Covering the TI99/4A and the Myarc 9640

MICROpendium

Volume 6 Number 6 July 1989 $2.00

Four-Card solitaire in BASIC

★ ★ ★

A better way to RESequence in XBASIC

★ ★ ★

More on polynomials in c99

The Portable TI — part 2

USING OVERLAYS IN XBASIC

The conclusion of a five-part series

THE BOX ABOVE WAS CREATED BY LOADING THE PICTURE OF "CHIP" THEN TYPING THE TITLE USING THE "301_LG" FONT FROM ONE OF THE FONT DISKS. FINALLY I TYPED THE BORDER AROUND IT. TOTAL TIME WAS ONLY MINUTES.

Review of Page Pro 99

ALSO INSIDE

- Continuation of CHARA1FIX character generator
- User-supported software additions
- Make your speech synthesizer read D/V80 files
- Appointment Scheduler runs out of Myarc Advanced BASIC

Super Savings

Tex-Comp Proudly Presents

Never Before Pricing

On Essential Software Packages for the Texas Instruments Home Computer

As part of its program of long-term support for the TI-99/4A user, Tex-Comp has purchased truck-load quantities of original TI Software that is essential to the serious & dedicated user.

In turn, Tex-Comp is passing the savings on to **YOU!**

All TI Software in this advertisement is brand new, original TI Product in factory-sealed packages and is sold with a full Texas Instruments warranty, which TI has publicly committed to.

Now is the time to buy **Key Software** at a fraction of its original cost.

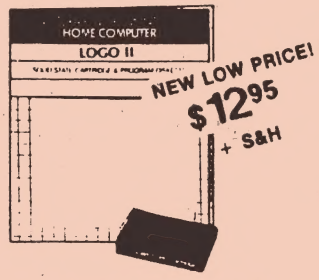
There may never be a better time than now to buy

TI Multi-Plan

Electronic Worksheet with many advanced features and built-in ease of use. Requires disk drive and controller, and 32K memory Expansion Unit. Printer and RS-232 Interface recommended. Cartridge and Disk.

TI Writer

This is a professional word processing system for the TI-99/4A. Provides the features and ease of use found in office systems. Requires disk drive. 32K Memory and Printer. Module and Disk.



Logo II

In use by educators throughout the country. Requires cassette or disk based system and 32K memory expansion.

Editor/Assembler

This is the **complete** version with manual, module, program disk and the disk version of Tombstone City as an example of assembly language programming. 32k and disk drive are required. This package will allow you to program the 99/4A in TMS 9900 Assembly Language and gives you access to all system features. Provides the fastest speed possible from the 16-bit processor!

Original TI-Extended Basic Still the BEST for Less!

A powerful, high-level programming language that expands the capability of your TI-99/4A Home Computer. Includes TI Extended BASIC module (36K bytes of preprogrammed memory) and Owner's reference manual.

Mini-Memory

This software cartridge adds memory to your system. Totals 14K of memory (6K of GROM, 4K of ROM, 4K of RAM). Mini Memory includes a built-in battery, permitting programs and data stored in RAM to be retained even if module is removed from console.

BONUS: FREE **Mini-Writer word processor (a $19.95 value) with Mini-Memory Purchase.**

SPECIALS:

SUPER EXTENDED BASIC.ExBasic + additional commands and graphics	$49.95
EXTENDED BASIC II PLUS BY Mechatronic..ExBasic plus 44 K Bytes of preprogrammed memory plus graphics.	$44.95
TI WRITER/MULTIPLAN UPGRADE DISK (with either)...	$ 1.00
TI LOGO WORKBOOK (with LOGO).....	$ 1.00
TEACH YOURSELF EXBASIC ON DISK (with any exbasic)	$ 1.00
"INTRO TO ASSEM. LANGUAGE" (with Ed/Assem).....	$ 1.00
TERMINAL EMULATOR II (reg 9.95) with any of above	$ 4.95

TEX★COMP

America's Number One TI computer retailer

P.O. Box 33084, Granada Hills, CA 91344

VISA & MASTERCARD HOLDERS CALL DIRECT

VISA MasterCard

(818) 366-6631

24 HOURS A DAY

7 Days a Week!

TERMS: All prices FOB Los Angeles. For fastest service use cashiers check or money order. Add 3% shipping and handling ($3.00 minimum). East of Mississippi, 4% (free shipping on all software orders over $100.00). COD to be paid by cash or certified check. All TI products are sold with the original manufacturer's guarantee only (sent on request). Prices and availability subject to change without notice. We reserve the right to limit quantities.

NOTE: Payment in full must accompany all orders. Credit Card, Company Check or Money Order for immediate shipment. Personal checks require up to 4 weeks to clear. California orders add 6½% sales tax.

Contents

MICROpendium

MICROpendium (ISSN 10432299) is published monthly for $20 per year by Burns-Koloen Communications Inc., 16606 Terrace Dr., Austin, TX 78728-1156. Second-class postage paid at Austin, Texas, and additional mailing offices. POSTMASTER: Send address changes to MICROpendium, P.O. Box 1343, Round Rock, TX 78680-1343.

All correspondence should be mailed to MICROpendium at P.O. Box 1343, Round Rock, TX 78680. We cannot take responsibility for unsolicited manuscripts but will give consideration to anything sent to the above address. Manuscripts will be returned only if a self-addressed stamped envelope is included.

Foreign subscriptions are $25.25 (Mexico); $27.50 (Canada) $25.00, surface mail to other countries; $37 airmail to other countries.

All editions of MICROpendium are mailed from the Round Rock (Texas) Post Office.

Mailing address: P.O. Box 1343, Round Rock TX 78680

Telephone: (512) 255-1512

CompuServe: 75156,3270
Delphi TI NET: MICROPENDIUM
GEnie: J.Koloen

John Koloen.....**Publisher**
Laura Burns.....**Editor**

Programming conventions

Here are some tips to help you when entering programs from MICROpendium:

1. All BASIC and Extended BASIC programs are run through Checksum, the numbers that follow exclamation at the end of each program line. Do not enter these numbers or exclamation points. Checksum was published in the October 1987 edition.
2. Long XBASIC lines are entered by inputting until the screen stops accepting characters, pressing Enter, pressing FCTN REDO, cursoring to the end of the line and continuing input.

Page Pro 99

Page Pro 99 is the remarkable new page-making program for your TI-99/4A or Myarc Geneve that does something no other publishing program can claim - let you see the page on the screen exactly as it will appear on paper, before it is printed.

Page Pro 99 is fast (written in assembly), easy to use (with on-screen prompts, a TI-Writer like editor, etc.), and powerful (with dozens of features, including many unique to this program). It will let you use standard TI-Artist pictures and fonts on your page. It will work on any system from a singe drive 99/4A to a Geneve with hard disk drives. It includes thorough documentation. It is very dependable and well-tested (as it should be after 2 years of development). Finally, it produces beautiful pages.

Because it is "what-you-see-is-what-you-get" it is easy-to-use. Because it includes numerous examples, a good collection of starter fonts and pictures, and an in-depth tutorial you can use it straight out of the box. Because it is by Asgard Software, it has the support that it, and you deserve. Who can ask for more?

PAGE PRO TIMES

VOL.1 NO.1 SUMMER 1989

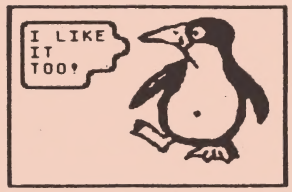
PAGE PRO 99 RELEASED!

Rockville, MD- Asgard Software demonstrated its long awaited Page Pro 99 at a press conference today to thunderous applause. After the preliminaries, the President (and chief bottle-washer) of Asgard, Chris Bobbitt, ran the program through its paces. Spectators were impressed with the program's graphics capabilities (you can have up to 28 pictures of any size on a page at once), the ability to see the page on the screen before printing it, its ease-of-use (it's completely "what-you-see-is-what you-get"), the multiple font features, and the line-drawing options.

Mr. Bobbitt did caution, "This isn't a full-scale desktop publisher, instead we like to call it a 'page-maker'", but most attendees felt the program would let the user do most anything that so-called desktop publishers could, and much more easily too. The general opinion from those who had a chance to use the program was that being able to see everything on the screen as it would appear on paper was a leap beyond anything else for the 99/4A or Geneve 9640.

When asked, attendees listed as reasons for purchasing the program many other things it offers including: the ability to use standard TI-Artist artwork, the ability to type in any direction, being able to print out the page at several dot densities, and also the ability to import TI-Writer files were all mentioned. Reviewers also liked its Myarc HFDC and 9640 compatibility, and the general stability of the program (which users speculate was because of its lengthy testing period).

Continued on Page 2



PROGRAM FEATURES

- 28 Pictures/page
- Large font and a small font on each page
- 2 linestyles/page
- See page on screen before printing
- Print in single, double or quad-densities
- Type up/down/left or right
- Import TI-Writer files
- Save/load page
- Uses TI-Artist Instances & Fonts
- And much more!
- Requires Disk, 32K XB or E/A modules

$27.95

PLUS $0.75 S&H
Canadians add $0.50
Airmail add $2.75

ASGARD SOFTWARE
BOX 10306
ROCKVILLE, MD 20850
(703)255-3085
MASTERCARD/VISA

Features:

- Paste up to 28 pictures of any size on a page
- Type in one large font and one small font per page
- Two linestyles can be used at once
- Print out pages in single, double or "quad" density
- Type in any direction - up/down/left or right
- Includes utilities for converting TI-Artist fonts and instances, as well as creating two column text
- Import/Export TI-Writer files, save and load pages
- Geneve specific version included
- Use TI-Writer keys for editing text. Place pictures wherever the cursor is
- Requires 32K, Disk, Epson compatible printer.

Asgard Software
P.O. Box 10306
Rockville, MD 20850
(703)255-3085

Only $24.95

U.S. please add $0.75 S&H
Canadians please add $1.25 S&H
Europe/Asia/Australia please add $4.00 S&H

Comments

We just ran out of space

You will notice that this month we've got a number of very long articles. All of them are installments in different series. I don't think this has ever happened to us before, where we had have more than one ongoing series at the same time. The result is that we weren't able to publish the screen output version of the Calendar programs nor the Myarc Q&A column. Unfortunately, we weren't even able to conclude the CHARA1FIX program by Wayne Stith, which we started last month. We were able to devote five pages to the program, but that still leaves more than 300 lines of code left for next month. We will finish it next month.

LOOKING AHEAD

In the next month or two we will be publishing a barebones terminal emulator program that supports XMODEM. We're going to do this because readers who use TEII, for example, can't download programs from many bulletin boards because TEII doesn't support XMODEM, a requirement for most boards. (CompuServe, GEnie and Delphi, for example, all require XMODEM to download programs.) Readers may input this program, log onto a bulletin board and, using the XMODEM feature, download a full-featured terminal program, such as Telco or Fast-Term, and be on their way. Incidentally, the program is written in assembly language is is about 400 lines long.

Also coming next month will be the start of a series of short articles on the p-code system, an article on Forth in 80 columns, and the start of a two-parter on standardizing printer functions.

ENCOURAGING MORE NEWSBYTES

This is directed at commercial and noncommercial software and hardware developers.

If you are trying to reach thousands of TI users, the best place to announce your products is in MICROpendium. There's no charge for a Newsbyte announcement, and we give this information a high priority. Announcements may be mailed to us, or posted to us on GEnie, CompuServe or Delphi (our ID's are listed on Page 4).

We often get inquiries from readers about where to obtain programs, particularly for Geneve programs. For those who distribute their programs by uploading them to bulletin boards, you should know that most of our readers do not belong to bulletin boards. In many cases, the only contact many of our readers have with the TI world is through MICROpendium. We encourage you to take a few minutes to drop us a line. If getting your product into the hands of users is important to you, you can't do better than by placing a free announcement in Newsbytes.

GENEVE UPDATE

Advanced BASIC is nearing completion of the debugging process, thanks to beta testing by scores of Geneve users. The most recent version, in late June, had only minor problems. Apparently none of the remaining problems affects the operation of the program. The most recent version runs out of Ver. 0.95 of the hard disk version of MDOS. Ver. 0.95, incidentally, isn't compatible with Myarc Disk Manager 5.

Jim Uzzell has released the first program we've seen to run out of Advanced BASIC. Called Appointment Scheduler, it allows the user to create an appointment calendar covering an entire year. The cost is $15. (See Newsbytes for details.)

THE SOURCE IS NO MORE

The Source telecommunications service has been acquired by CompuServe. Source subscribers were notified in July of the situation. The transfer is effective as of Aug. 1. Source users received a $20 credit for usage time on CompuServe as a result of the discontinuation of The Source.

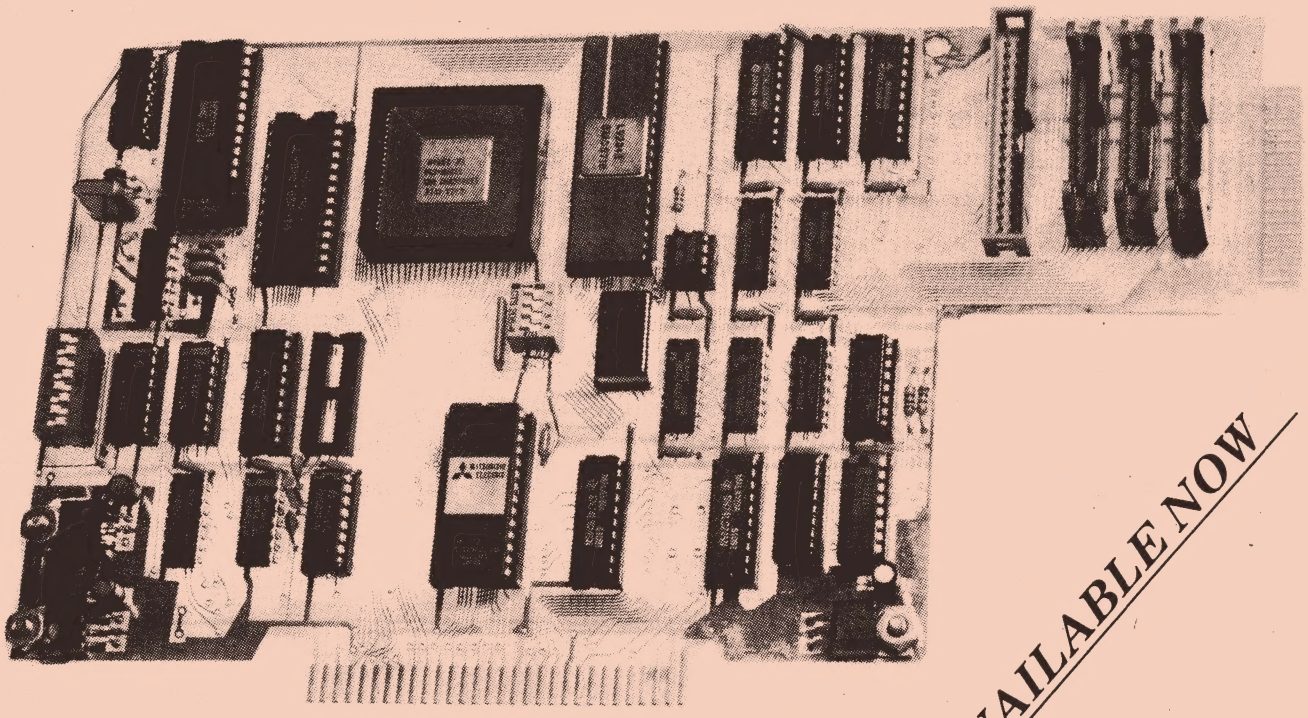
Unlike CompuServe members who pay no minimum monthly fee, Source members have been required to pay a minimum monthly fee and, according to CompuServe, will continue to do so as CompuServe members. Source members can check with the Sysops (TI-FORUM) on CompuServe to clarify this.

DON'T EXPECT MORE TENEX CATALOGS

Aside from filling orders from its current catalogs, I don't expect to see any more catalogs of TI products from Tenex Computer Express.

—JK

THE MYARC Hard & Floppy Disk Controller with Streamer Tape Backup Support



AVAILABLE NOW

Features include:

- Hard drive transfer rate of 5Mbit per second, for speed comparable to an external RAM disk card
- Interfaces with standard, off the shelf, hard, floppy and streamer tape drives
- Built-in real time clock, for time and date stamping of files
- Supports up to four 5 ¼" and/or 3 ½" floppy drives, mix or match
- All disk formats, SS/SD (90K), DS/DD (320/360K) and DS/QD (640/720K) supported
- MYARC Disk Manager V, the most intuitive and user friendly manager available
- One year limited warranty, 12 months parts, 6 months labor, is standard, an optional two, three or four year extended warranty is available

MYARC, Inc., P.O. Box 140, Basking Ridge, New Jersey 07920, Phone (205) 854-5843

Feedback

NX-1000 problems

As a followup on my article (March 1989) I have found from reading several newsletters and from experience with the problems of a local user group member that some TI99/4A owners are having trouble with their NX-1000 printers.

It appears that the later versions of the NX-1000 are not compatible with the TI99/4A because of a problem in an EPROM chip in the printer. Although the self test works fine and the printer appears to operate with other computers, nothing is printed from the TI99/4A. The problem seems to be in just certain versions of the printers' EPROM. One version I have found to be bad is version 1.5. My NX-1000 has a version 1.31 and works fine.

The version number is printed when the self test is performed and is also written on the EPROM itself, which is located next to the dip switches under the cover. From what I have been able to determine, Star is providing free of charge an updated EPROM to correct this problem.

Many computer stores are authorized Star repair centers. Otherwise, you must call Star Technical Support at 1-900-860-9104. The charge for calling this 900 number is $2 for the first minute and $1 for each additional minute, according to what I was told when I called the 800 number. In calling the 800 number (800-537-8270), I was told that the 800 number is for authorized service centers only. Their old technical number is disconnected, and, in calling their general information number (714-768-7203) I was told that I must call the 900 number if I want service!

Anyway, I thought I would pass this information along in case some TI owners are having similar trouble with their NX-1000s.

Gary Cox
President, Mid-South 99 User Group
Memphis, Tennessee

Thanks for help

I want to send thanks for all the replies I received for my letter to Reader to Reader (May 1989). Of course everyone will be receiving a personal thanks by return letter.

I want to mention that part of my letter dealing with copying single-sided disk to double-sided disk was solved by information received from C. Morrison of Worcester, Massachusetts. He put me on to Quick-Copier II, from Quality 99 Software at 1884 Columbia Rd. #1021, Washington, DC 20009-5161, telephone number (202) 667-3574. This program will copy disks and maintain the format of the target disk. It does a full single disk in only three passes; the instructions say it will do it in two passes with Mini-Memory, but I cannot get it to load with that cartridge. I use Extended BASIC. It sells for $9.95 plus $4.50 handling. It is a time saver for me.

Ray Russell
Weatherford, Texas

Geographical basis for Myarc support?

Letters in support of Myarc support, published in MICROpendium, have one thing in common — all — *all* — are from the USA. The initial complaint which triggered this pro-Myarc avalanche came from Canada.

I am aware of similar complaints from Sweden and Australia. Overseas supporters are having a bad time of it.

Perhaps MICROpendium can obtain letters of support for Myarc from outside the USA?

Stephen Shaw
Stockport, England

Mail problem

I desperately need the help of the TI community. Would anyone who has sent me a donation for my 1989 Valentine after Feb. 28 who has still not received a reply from me, please immediately send me a letter, clearly stating the following information:

1. Name and current address (including country, if not in the USA).
2. The exact date, or close approximation, when originally mailed.
3. Amount of donation.
4. Manner in which it was sent (cash, check or money order).
5. Address to which the donation was sent (if known).
6. If a letter containing a donation was returned to the sender, please include the original envelope (if the sender still has it).

The post office is aware that some mail is either being mishandled or stolen and is investigating claims.

Upon receiving a letter with the above information in it, I will immediately mail a post card back, acknowledging receipt. Mail sent directly to my new address (below) has not been affected by this situation and will reach me with no difficulty.

Persons assisting with this investigation will not be called on to testify in the event this case results in an arrest.

Ray Kazmer
8614 Foothill Blvd., #221
Sunland, CA 91040

Program listings and chess

Just a few lines in response (albeit a little late) to your statement about program listings in MICROpendium printed via the 24-pin dot matrix printer instead of the old way; I sure appreciate the better legibility of this newer way, as I used to bypass typing in programs from the old format even if I badly wanted to make use of them — just couldn't decipher some tops and tails with accuracy previously. Good show!

Also, your mention of a chess program for the 4A, based on the Sargon algorithm — I've been mystified as to why no such program was developed for us beyond what Texas Instruments put out. Certainly such a game shouldn't be ignored by programmers — a jewel among games of logic, a veritable evergreen with worldwide recognition for centuries. It's been a real shame. At least Asgard helped out some with its Beyond Video Chess, which benefits players immensely in enabling them to use disk I/O of games and to use joysticks in comfort rather than keyboard only. But I hope that if there is such a program for the TI that it will be much more powerful than the one TI put out in its module. That item had a lot of bells and whistles that could have been eliminated to give us a much stronger program, which is what chess lovers really enjoy.

Charles Poulin
Euclid, Ohio

TI INTERNATIONAL EXPO 89

Sponsored by Mid-Atlantic Ninety-Niners

Saturday, September 16, 1989

VENDORS — USER GROUPS — DEMOS — SPEECHES — NEW PRODUCTS
ALL FOR THE 99/4A AND 9640 COMPUTERS.

Howard Johnson Inn
5821 Richmond Highway
Alexandria, Virginia
703-329-1400

General Admission: $4.00
Advance Sales: $3.00
Buffet, advance sales: $15.00

Show Time: 9:00 am — 5:00 pm
Pre-show reception: 6:00 pm Friday
Buffet: 6:00 pm Saturday

Vendor Booth Rates: $50.00 first table
$15.00 each additional table
Contact us for additional details.

SPEND A FAMILY VACATION IN THE BEAUTIFUL NATION'S CAPITAL

Hotel Rates: $55.00 + tax. Avoid a sell out! You must reserve your room by August 25 to obtain the special show rates. Call Howard Johnson now! International attendees, contact us directly.

Call or write for additional details, vendor information or advance sales:

TI International Expo 89, P.O. Box 4005, Rockville, MD 20850

Call the EXPO 89 Info Line: 301-340-7179

Delphi: TELEDATA - CIS 74405,1207

★★★ TI NET ★★★

The Texas Instruments International Network

WE DON'T JUST "SUPPORT" THE 99/4A AND 9640 GENEVE
WE ARE *THE* SUPPORT FOR YOUR SYSTEM!!

COMPARE COSTS:	DELPHI	CompuServe #	GEnie +
1200 baud	$7.20/4.80*	$12.50	$6.00
2400 baud	$7.20/4.80*	$12.50	$10.00
Canada nodes	$7.20/4.80**	None	$13.20/14.40 ++

* $4.80 — Advantage plan rate
** In most major Canadian cities, U.S. Dollars. Advantage plan available
\# CompuServe Surcharge, add $1.50/mo. Network charges included - .50/hr
\+ Some remote access cities have an additional $2.00/hr. surcharge.
++ GEnie 1200/2400 baud charges
All costs are per hour during non-prime connect time.
Delphi Available on the Tymnet and Telenet networks

THE FEATURES!

TI NET has the usual Databases, Message areas and Conference facilities but TI NET has much more the others just don't have!

* Electronic Shopping — features online purchasing for your system.
* Interactive Polling — allows you to voice your opinions.
* Classified Advertising — Interactive ads to sell used products.
* Online Graphics — supported by the TELCO program.
* Protocols — YMODEM support for faster up/down loads.
* Surcharge Software — Pays you $$ for software uploaded you designate
* Magazine ordering — Order 99/4A and 9640 magazines online.
* Much more than space can permit!

UNANSWERED QUESTIONS?

You won't have any as a member of TI NET. If your 99/4A or 9640 related question can't be answered on TI NET, it can't be answered at all!

Communicate directly with the designers and developers of many of the 99/4A and 9640 products you use daily.

IT'S GUARANTEED!!

Join by August 1, 1989 — leave one question concerning ANY aspect of your system or software by August 31, 1989, and it's guaranteed you'll receive a reply to your queston. If not, you'll receive one FREE hour of standard connect time on TI NET when you let the System Manager know.

ONLINE SIGN UP OFFER

Dial **1-800-365-4636** or **617-576-2981**
Press Return twice. At PASSWORD, type TISPECIAL and press Return. From Canada: dial direct. At USERNAME type JOINDELPHI and Return.

Delphi: General Videotex Corp.
3 Blackstone St.
Cambridge, MA 02139

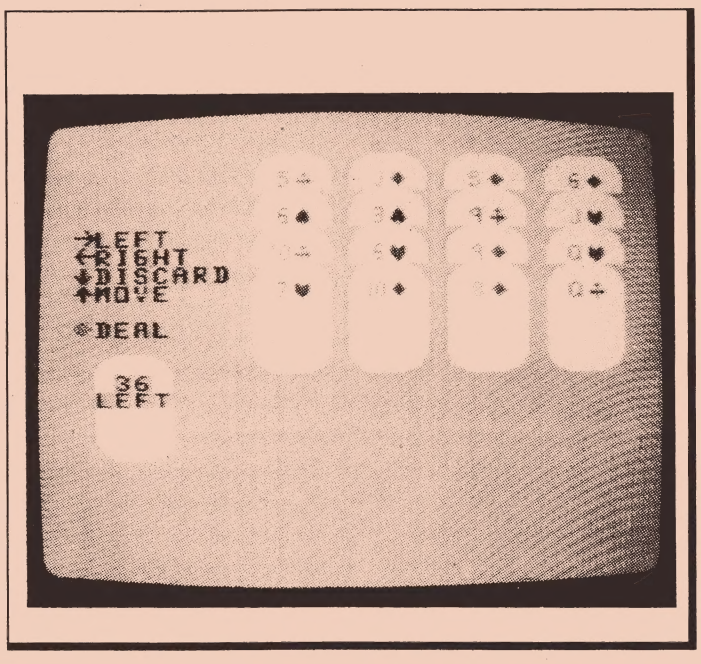
TI NET: P.O. Box 244
Lorton VA 22079

BASIC

It's time for fourcard solitaire

By REGENA

It's time for another solitaire card game. This month's program is written in TI BASIC for the card game "Fourcard." The computer is used to deal the cards randomly and display the cards. The computer does not allow you to make an illegal move.



Four cards are dealt face up at the top of four columns. The object of the game is to discard or move cards as more cards are dealt so that eventually the four aces are at the tops of the columns. Only four cards are "playable" at any time — the cards showing completely at the bottoms of the columns.

The asterisk at the top of the column indicates which bottom card you want to select. Use the arrow keys (S and D without pressing the FCTN key) to move the asterisk left or right. When any of the cards showing have matching suits, you may discard or remove the card(s) with the lower numerical value. Ace is high. To discard, move the asterisk over the appropriate column, then press the down arrow key (X without FCTN). The card will be removed, and if there are more cards in that column, the new bottom card will now show.

If you remove cards in the top row, a blank space will be created (an empty column). Any other bottom card may be moved to that top vacant space. Move the asterisk above the column you want — the one with the card you want moved — and press the top arrow key (E without FCTN). The card will move from the bottom of the column selected to the space in the top row.

When you have discarded or moved all the cards you can, press the ENTER key to deal four more new cards. Continue the process of discarding, moving and dealing, and try to get all the aces to the top row. When all four aces are in the top row, press ENTER and the game ends. If you try to deal more than nine rows of cards in any column, the game will end as a loss. The card deck on the left of the screen indicates how many cards are left in the deck to be dealt.

As you play the game you will learn different strategies. For example, you may not always want to move the ace to the top spot immediately — another move may free two cards of the same suit and allow discarding more cards before moving the ace up. I have been able to get the four aces in place with 20 cards left in the deck (lucky shuffle). Have fun trying to beat that!

EXPLANATION OF THE PROGRAM

Line 120 DIMensions arrays used in the program. CARD(52) are the possible 52 cards in the deck. T(r,c,1) is the suit of the card in row r and column c. T(r,c,2) is the number of the card in row r and column c. R(4) keeps track of which row in each of the four columns has the playable card.

Lines 130-480 print the title screen and brief instructions while graphic characters and colors are defined. Remember, if you have trouble running the program, the most likely place for a typing error is in the DATA statements of Lines 410-470. These lines define graphic characters for the cards and numbers and suits on the cards.

Lines 480-500 wait for you to press the ENTER key before starting the game. Line 510 clears the screen, then Lines 520-610 initialize variables for "shuffling" the cards and starting a new game.

Lines 480-500 wait for you to press the ENTER key before starting the game. Line 510 clears the screen, then Lines 520-610 initialize variables for "shuffling" the cards and starting a new game.

Lines 620-730 clear the screen then print the playing screen with the reminder arrows and the deck showing how many cards are left to be dealt. Lines 740-830 deal the first four cards. The variables J and K are used as coordinates of the top left corner of a card for drawing or removing a card.

Line 840 starts with A=1, where A designates the number of the column indicated by the asterisk. AA is the graphics column coordinate corresponding to A. Lines 860-890 blink the asterisk while waiting for the player to press a key.

Lines 900-950 are the procedure if the left arrow key is pressed. The asterisk moves one column to the left, or it stays at the left-most column. Lines 960-1010 move the asterisk one column to the right, or keep it at the right-most column. A "beep" is sounded whenever the asterisk is moved.

Lines 1020-1180 contain the procedure for the up arrow, or moving a card from the bottom of a column to an empty column. Lines 1040-1070 make sure there is an empty space in the top row. Line 1080 makes sure the card to be moved is not already in the top row. Lines 1090-1160 move the card to the top row, and Line 1170 calls the subroutine to erase the card from the bottom.

Lines 1190-1300 containg the procedure for discarding. Line 1210 makes sure the asterisk is not over an empty column. Lines 1220-1290 erase a card only after checking to make sure a card in another column has the same suit and is of higher value.

Lines 1310-1560 contain the procedure for dealing four more cards when the ENTER key is pressed. First, Line 1330 checks to see if more cards are available; otherwise the game ends. Lines 1340-1360 make sure there are not nine cards in a column (or the

(See Page 11)

REGENA ON BASIC —

(Continued from Page 10)

game ends.

Lines 1370-1410 check to see if the four aces are at the top of the columns. If the four aces are in place, the game will end. If you prefer to let the game continue, you may delete Line 1410, which goes to the end.

Lines 1420-1490 deal the four cards in the appropriate positions at the bottom of each column. Line 1500 calculates how many cards are left in the deck, and Lines 1510-1550 print how many cards are left.

Lines 1570-1840 are the subroutine to choose a card randomly and then draw it. The subroutine from Line 1690 to 1840 is also used to draw a card when it is moved to the top of a column. Line 1580 is RANDOMIZE to make the game different each time.

Lines 1590-1620 choose the cards if they are the last four cards to be dealt. Rather than taking the time to find them with random selection, the last four are just selected from those left. Unfortunately, they will be in order numerically in the suits hearts, diamonds, spades and clubs. However, waiting for these cards to be selected randomly is usually noticeably slower. If you prefer the random selection, delete Lines 1590-1620.

Other cards are selected randomly in Line 1630. CC is a number from 1 to 52. Lines 1640-1650 make sure that card has not previously been chosen. When the card is selected, CARD(CC) is set equal to 1. If it has not been previously selected, CARD(CC)=0. Lines 1660-1680 determine the suit and number of the card. The first 13 numbers are hearts, then diamonds, then spades, then clubs. The cards are actually numbered 1 through 13 but will show 2 through 10, then J, Q, K, A. I did this so that when the numbers are compared, the Ace will be the highest number.

Lines 1690-1840 draw the card using the position of the upper left corner with coordinates J,K. Lines 1710-1780 draw the white card. Line 1790 draws the symbol for the suit. Lines 1800-1830 draw the number red or black depending on the suit. The suit symbols are 118, 119, 120 and 121, where 118 and 119 are in the red character set and 120 and 121 are in the black set.

Lines 1850-2040 are the subroutine to erase a card. J is the row coordinate, and AA is the same column as the asterisk. Lines 1890-1990 remove the card and draw the card underneath if the card is not on the top row. If the card is on the top row, the card is erased with Lines 2000-2040. When a card is discarded, the row number is decreased in Line 1980 or 2030.

Lines 2050-2080 are the subroutine to print a message M$ on the screen at row J and column K+1 without scrolling.

Lines 2090-2180 print the message when the game is over and whether you won or lost. Lines 2190-2240 present the option to play again and wait for you to press the Y key for Yes or N key for No. Lines 2250-2260 clear the screen and end the program.

Remember as you are typing this program in from the listing that the ! and numbers at the end of each line are not to be typed. These numbers indicate checksum numbers when you use Tom Freeman's Checksum program as an aid to typing in published programs (see note on Contents page of this issue).

If you prefer to save typing effort, you may have a copy of this program by sending $4 to *REGENA, 918 S. Cedar Knolls West, Cedar City, UT 84720. Be sure to specify that you need the TI version of Fourcard and whether you want cassette or diskette.*

Fourcard Solitaire

```
100 REM FOURCARD !048
110 REM BY REGENA !071
120 DIM CARD(52),T(9,4,2),R(
4)!133
130 CALL CLEAR !209
140 PRINT TAB(5);"** FOURCAR
D **" !202
150 CALL CHAR(60,"080402FF02
0408")!148
160 PRINT : :"FOUR CARDS ARE
 DEALT.  IF" !046
170 CALL CHAR(61,"102040FF40
201")!086
180 PRINT "ANY SUITS MATCH,
YOU MAY" !248
190 CALL CHAR(62,"0010101092
54381")!162
200 PRINT "DISCARD THE LOWES
T CARD BY" !107
210 CALL CHAR(64,"0010385492
10101")!164
220 PRINT "USING THE ARROW K
EYS TO MOVE" !039
230 CALL CHAR(136,"00183C7E7
E3C18")!247
240 PRINT "OVER THE RIGHT CO
LUMN, THEN" !194
250 CALL COLOR(9,16,1)!233
260 PRINT "PRESS THE > KEY."
 !150
270 CALL COLOR(10,7,16)!024
280 PRINT :"TO DEAL 4 MORE C
ARDS, PRESS" !025
290 CALL COLOR(14,11,1)!017
300 PRINT "THE ";CHR$(136);"
ENTER KEY." !254
310 CALL COLOR(11,7,16)!025
320 PRINT :"PRESS @ TO MOVE
A BOTTOM" !141
330 CALL COLOR(12,2,16)!021
340 PRINT "CARD TO AN EMPTY
TOP SPOT." !095
350 CALL COLOR(13,2,16)!022
360 PRINT :"GET ALL FOUR ACE
S ON TOP!": : :!178
370 FOR C=96 TO 134 !218
380 READ C$ !254
390 CALL CHAR(C,C$)!081
400 NEXT C !217
410 DATA 00030F1F3F3F7F7F,FF
FFFFFFFFFFFFF,00C0F0F8FCFCF
EFE,FEFEFCFCF8F0C,7F7F3F3F1F
0F03,,,0 !042
420 DATA 1824040408103C,3C04
0418040438,202424243E0404,3C
4038040424 38,1C20203C24241C,
3C0404080810 1 !005
430 DATA 18242418242418,1C24
241C040404,8C92929292928C,04
040404042418,38444444444C34,
242830282824 24 !085
440 DATA 18242424243C2424,0,
22777F7F7F3E1C08,081C3E7F3E1
C08,081C3E3E7F7F3608,081C082
A7F2A0808 !084
450 DATA 1824040408103C,3C04
0418040438,202424243E0404,3C
4038040424 38,1C20203C24241C,
3C0404080810 1 !005
460 DATA 18242418242418,1C24
241C040404,8C92929292928C,04
```

(See Page 12)

REGENA ON BASIC—

(Continued from Page 11)

```
040404042418,38444444444C34,
24283028282424 !085
470 DATA 18242424243C2424 !1
74
480 PRINT :"PRESS ";CHR$(136
);"ENTER TO START.";!130
490 CALL KEY(0,KEY,S)!089
500 IF KEY<>13 THEN 490 !126
510 CALL CLEAR !209
520 PRINT "SHUFFLING DECK...
" !219
530 FLAG=0 !209
540 FOR C=1 TO 52 !105
550 CARD(C)=0 !129
560 NEXT C !217
570 FOR C=1 TO 4 !053
580 FOR D=1 TO 2 !052
590 T(1,C,D)=0 !095
600 NEXT D !218
610 NEXT C !217
620 CALL CLEAR !209
630 PRINT "<LEFT" !207
640 PRINT "=RIGHT" !036
650 PRINT ">DISCARD" !163
660 PRINT "@MOVE" !223
670 PRINT :CHR$(136);"DEAL"
!142
680 PRINT :" `aab" !193
690 PRINT " a48a" !182
700 PRINT " LEFT" !179
710 PRINT " aaaa" !012
720 PRINT " aaaa" !012
730 PRINT " daac": : : : : :
!079
740 J=2 !003
750 K=12 !054
760 DECK=48 !011
770 FOR D=1 TO 4 !054
780 GOSUB 1580 !130
790 T(1,D,1)=SUIT !098
800 T(1,D,2)=N !108
810 R(D)=1 !187
820 K=K+5 !019
830 NEXT D !218
840 A=1 !249
850 AA=8+5*A !004
860 CALL KEY(0,KEY,S)!089
870 CALL HCHAR(1,AA,42)!079
880 CALL HCHAR(1,AA,32)!078
890 IF S<1 THEN 860 !104
900 IF KEY<>83 THEN 960 !093
910 REM MOVE LEFT !092
920 IF A=1 THEN 860 !085
930 CALL SOUND(100,1492,2)!1
85
940 A=A-1 !252
950 GOTO 850 !164
960 IF KEY<>68 THEN 1020 !15
6
970 REM MOVE RIGHT !175
980 IF A=4 THEN 860 !088
990 CALL SOUND(100,1492,2)!1
85
1000 A=A+1 !251
1010 GOTO 850 !164
1020 IF KEY<>69 THEN 1190 !0
72
1030 REM MOVE UP !214
1040 FOR B=1 TO 4 !052
1050 IF R(B)=0 THEN 1080 !24
1
1060 NEXT B !216
1070 GOTO 860 !174
1080 IF R(A)=1 THEN 860 !020
1090 J=2 !003
1100 K=7+5*B !205
1110 SUIT=T(R(A),A,1)!101
1120 N=T(R(A),A,2)!111
1130 R(B)=1 !185
1140 T(1,B,1)=SUIT !096
1150 T(1,B,2)=N !106
1160 GOSUB 1700 !250
1170 GOSUB 1870 !165
1180 GOTO 860 !174
1190 IF KEY<>88 THEN 1310 !1
94
1200 REM DISCARD !212
1210 IF R(A)=0 THEN 860 !019
1220 FOR B=1 TO 4 !052
1230 IF R(B)=0 THEN 1290 !19
6
1240 IF A=B THEN 1290 !077
1250 IF T(R(B),B,1)<>T(R(A),
A,1)THEN 1290 !081
1260 IF T(R(A),A,2)>T(R(B),B
,2)THEN 1290 !148
1270 GOSUB 1870 !165
1280 B=6 !255
1290 NEXT B !216
1300 GOTO 860 !174
1310 IF KEY<>13 THEN 860 !24
2
1320 REM DEAL !240
1330 IF DECK=0 THEN 2100 !00
7
1340 FOR B=1 TO 4 !052
1350 IF R(B)=9 THEN 2100 !25
0
1360 NEXT B !216
1370 FOR BB=1 TO 4 !118
1380 IF T(1,BB,2)<>13 THEN 1
420 !219
1390 NEXT BB !026
1400 FLAG=1 !210
1410 GOTO 2100 !139
1420 FOR B=1 TO 4 !052
1430 R(B)=R(B)+1 !123
1440 J=2*R(B)!199
1450 K=7+5*B !205
1460 GOSUB 1580 !130
1470 T(R(B),B,1)=SUIT !103
1480 T(R(B),B,2)=N !113
1490 NEXT B !216
1500 DECK=DECK-4 !171
1510 D$=" "&STR$(DECK)!037
1520 M$=SEG$(D$,LEN(D$)-1,2)
!163
1530 J=14 !055
1540 K=4 !006
1550 GOSUB 2050 !090
1560 GOTO 860 !174
1570 REM DEAL CARD !042
1580 RANDOMIZE !149
1590 IF DECK>4 THEN 1630 !05
3
1600 FOR CC=1 TO 52 !172
1610 IF CARD(CC)=0 THEN 1650
 !057
1620 NEXT CC !028
1630 CC=INT(52*RND)+1 !006
1640 IF CARD(CC)<>0 THEN 163
0 !230
1650 CARD(CC)=1 !197
1660 SUIT=INT((CC-1)/13)!224
1670 N=CC-13*SUIT !138
1680 SUIT=SUIT+1 !003
1690 REM DRAW CARD !066
1700 CALL SOUND(100,592,2)!1
36
1710 CALL VCHAR(J,K,96)!127
1720 CALL VCHAR(J+1,K,97,4)!
235
1730 CALL VCHAR(J+5,K,100)!0
97
1740 CALL VCHAR(J,K+1,97,6)!
237
1750 CALL VCHAR(J,K+2,97,6)!
238
1760 CALL VCHAR(J,K+3,98)!06
2
1770 CALL VCHAR(J+1,K+3,97,4
)!168
1780 CALL VCHAR(J+5,K+3,99)!
254
1790 CALL HCHAR(J+1,K+2,117+
SUIT)!025
```

(See Page 13)

REGENA ON BASIC—

(Continued from Page 12)

```
1800 IF SUIT>2 THEN 1830 !04
2
1810 CALL HCHAR(J+1,K+1,103+
N)!028
1820 GOTO 1840 !134
1830 CALL HCHAR(J+1,K+1,121+
N)!028
1840 RETURN !136
1850 REM DISCARD !212
1860 REM ERASE CARD !132
1870 J=R(A)*2 !198
1880 CALL SOUND(100,440,2)!1
28
1890 IF R(A)=1 THEN 2000 !14
0
1900 CALL HCHAR(J,AA-1,97,4)
!021
1910 CALL HCHAR(J+1,AA-1,97,
4)!208
1920 CALL HCHAR(J+2,AA-1,97,
4)!209
1930 CALL HCHAR(J+3,AA-1,100
)!068
1940 CALL HCHAR(J+3,AA,97,2)
!020
1950 CALL HCHAR(J+3,AA+2,99)
!036
1960 CALL HCHAR(J+4,AA-1,32,
4)!200
1970 CALL HCHAR(J+5,AA-1,32,
4)!201
1980 R(A)=R(A)-1 !122
1990 RETURN !136
2000 FOR BB=AA-1 TO AA+2 !25
1
2010 CALL VCHAR(J,BB,32,6)!0
96
2020 NEXT BB !026
2030 R(A)=0 !183
2040 RETURN !136
2050 FOR BB=1 TO LEN(M$)!044
2060 CALL HCHAR(J,K+BB,ASC(S
EG$(M$,BB,1)))!096
2070 NEXT BB !026
2080 RETURN !136
2090 REM GAME OVER !080
2100 M$="GAME OVER" !117
2110 J=19 !060
2120 K=2 !004
2130 GOSUB 2050 !090
2140 M$="YOU LOST" !093
2150 IF FLAG<>1 THEN 2170 !0
18
2160 M$="YOU WON!" !048
2170 J=21 !053
2180 GOSUB 2050 !090
2190 M$="PLAY AGAIN? (Y/N)"
!067
2200 J=24 !056
2210 GOSUB 2050 !090
2220 CALL KEY(0,KEY,S)!089
2230 IF (KEY=89)+(KEY=121)TH
EN 510 !127
2240 IF (KEY<>78)*(KEY<>110)
THEN 2220 !180
2250 CALL CLEAR !209
2260 END !139
```

EXTENDED BASIC

Line number crunching

By JERRY L. STERN

I don't like the RESequence command in TI Extended BASIC. Most of the language is great, certainly one of the most powerful versions of BASIC available today. But I learned BASIC on an antique mechanical teletype machine. The teletype was connected to a time-shared mainframe computer, and printed out everything on enormous rolls of coarse yellow paper with chunks of wooden knotholes still in it. There were no commands for any kind of fancy printout, just print a line, advance a line, print a line. The fancy print controls consisted of the backspace command. Period.

The language was crude, but it had a few (very few) redeeming features. The RESequence command could start in the middle of the program and leave the beginning alone. When I wanted to insert a new section of code in the middle of a program, I could respace the portion I needed without messing up the line numbers to the rest of the program.

The Extended BASIC and console BASIC RESequence, or RES, command is not that flexible. The entire program is always renumbered. The same increment is always used for line spacing throughout the program. Any subprogram that was deliberately numbered with a nice high set of line numbers is always moved down to the low number regions, where it gets all mixed up with the main program.

There is only one advantage to using either high or low numbers for program lines in TI BASIC — it makes the program easier to read and debug. There is no speed advantage, and no memory is saved by low line numbers. Using smoothly spaced lines makes the program more readable, and using high numbers for subprograms helps keep the program ready for adding new features by merging in additional subprograms later on.

The TI RES command handles bad line numbers in a poor fashion. Just because we may have lapsed for a moment into being human, and left out a line number that another line was supposed to branch to, is no reason to be told to "GOTO 32767." Is that a place? Do computers think we would suffer if we went there?

It is possible to write our own program to resequence programs in Extended BASIC. By saving a program in MERGE format, we can store the program in a format readable by another program. The format of the merge file is not obvious, but we will need to understand only a few small parts of the format for this application. First, line number storage at the beginnings of lines and in the middle of lines. Next, the storage of remarks lines, and finally, a little bit of background about the file end marks.

Extended BASIC program lines are stored in a shorthand form, with "tokens" representing each command name. There is a token number assigned to each Extended BASIC reserved word. Any program line is represented as a string of numbers.

(See Page 14)

EXTENDED BASIC—

(Continued from Page 13)

100 PRINT
is: 0 100 156 0
110 RANDOMIZE
is 0 110 149 0
120 GOTO 400
is 0 120 134 201 1 144 0
400 STOP
is 1 144 152 0

Line numbers are two bytes long. To convert back to the original line number, the first byte is multiplied by 256, and added to the second byte. In the middle of a line, the two bytes are preceeded by the token 201 as identification.

Admittedly, our RESequence II program, RESEQ2 for short, will run more slowly than does RES. So, we had better make it worthwhile to run it. We'll need a wish list.

WANTED

1. The ability to space different portions of the program with different increments.

2. The option of leaving the high end numbers, for DATA statements and subprograms, at their original locations.

3. A printout of line numbers before and after the resequence.

4. Better treatment for undefined line numbers than just "GOTO 32767."

RESEQ2 works by reading the original program, in MERGE format, twice. The first time through, RESEQ2 makes a list of all the line numbers in use, and displays the lowest and highest line numbers, and the starting line number of the first subprogram. Because Extended BASIC will not allow any main program lines after that first subprogram line, that is a natural dividing point for the program. I generally leave my subprogram with the high line numbers they were originally written with. That makes them easier to reuse in other projects.

After creating a line number conversion table, RESEQ2 reads the file again, and searches for all the line numbers. Each line number is converted to the newly assigned number, and the new set of lines is saved in a new merge file.

Let's start at the beginning. On line 120 space is saved for the line number table as the matrix, or two-dimensional array, A(1,300). A(0,X) will be the old numbers, and A(1,X) will be the new numbers. The choice of 300 lines is arbitrary. You may use a larger number if you think your programs will get that long. There is memory to spare, even with just the 32K memory card.

The DEF statements on lines 130 and 140 convert crunch or merge format line numbers to and from our normal base ten notation. Including them here makes these formulas less difficult to include in the operations to follow. DEF LN(T$) converts FROM crunch format, and DEF CR$(S) converts TO crunch format.

The code searches each line for other line numbers. A line number can be recognized within a line by searching for the ASCII character #201. The two bytes that follow that character are a line number in base 256.

Next, the file names are determined. The defaults of DSK1.TEMP and DSK1.NEWTEMP may be changed if you like other names.

Lines 200-260 read the original, or source file, for the first time, and convert the first two bytes of each line into the line numbers for both the new and old halves of the conversion table. Each line is checked until the first subprogram is found. That line will have the token for SUB immediately after the line number. That is, the command SUB will be represented in the third byte of the line by the token number 161.

When reading a disk file, usually the end of file command (EOF) is used to check that the program is not allowed to read past the end of the file. However, the last line of a merge format program file is not a program line, but the file end signal of the number 255 listed twice. RESEQ2 checks for the end of the file by looking for that code instead of the standard end of file code. When that signal is found, the file is set back to its beginning.

RESTORE #1

This statement allows the program to read the program file a second time without the need to close and reopen the file.

Next, the program asks for instructions on how to renumber the program lines. If no choice is given for a first line, 100 will be used as a default. An increment is requested next, and the line number to use that increment until. That limit does not need to be a real line number; it can be any number between the last line number to be changed and the first number to be either left alone, or changed with the next increment value.

As the program receives the new line numbering instructions, it updates the line number table. When it reaches the stop point requested, it asks again for a new increment. If ENTER is pressed by itself at this point, RESEQ2 will stop making changes and go on to the next step. If the user of the program doesn't plan carefully, and asks for a 50-line program to be renumbered with an increment of 1000, the line numbers will become larger than the maximum allowed value of 32767. If RESEQ2 discovers this has happened, it will stop at this point and reset the new line number table as being equal to the old, and then start the process of asking for renumbering instructions again. This time, try more reasonable numbers.

Line 430 allows the printout of the altered line number table. If no printer name is given, this section will be skipped, and the process of converting the program will begin.

For every line, RESEQ2 must perform several actions. Each line begins with an old line number. That number is converted to the new number assigned in the line number table. This is not done by searching for the old number, but instead is done by assigning the first new line number to the first new line, the second to the second, and so on.

Next, the code searches each line for other line numbers. A line number can be recognized within a line by searching for the ASCII character #201. The two bytes that follow that character are a line number inbase 256. The earliest position any line number could take within the body of the line would be immediately after the first command, for example, GOTO 200. That places the #201 code in the fourth byte of the line, so RESEQ2 will begin its search in that spot.

For each line number found in the body

(See Page 15)

EXTENDED BASIC—

(Continued from Page 15)

of a line, RESEQ2 searches the table of old line numbers for that number, and then replaces the number in the line with the two-byte code for the new line number. If there is no matching old number, there is a reference error in the program; an instruction has been written to send program control to a non-existent line. Rather than just replace the old line number with "32767," RESEQ2 will add a comment, or tail remark to the end of the line.

If an old line reads — 200 ON X GO SUB 300,350,400 — and there is no old line number 350, then the new line would read like this:

240 ON X GOSUB 310,32767,380 ! REFERENCE 340 TO 350

That means that the old line number was positioned in the program between the new lines 340 and 350. The old line 350 was probably a remark statement that was deleted during revision. There should never have been a branch to a comment statement in the first place, but the tail remark will still help repair the damage.

After making the swap of the new line number for the old, RESEQ2 goes back to three bytes past the position where it found the first old number, and searches again for the next old number. It will repeat this search, replace and search routine until the POS command reports that there are no more characters #201 in the line. Finally, the new line may be written to the new program file, including the new tail remark if it needed one, and the end of line signal, an ASCII code zero.

When all the lines have been converted, RESEQ2 prints one additional line to the new file, the line 255, 255. Those two ASCII characters signal the end of the merge file to Extended BASIC.

After running the program, the new file will be in merge format. To convert it back to program format, type:

NEW
MERGE DSK1.NEWTEMP
SAVE DSK1.NEWNAME

Well, that fixes one of the few annoying parts of Extended BASIC. There are some other features of other BASICs that would be a help to us in our programming. I think I had better get to work on converting them, too.

RESEQ2

```
100 ! RESEQ2 !085
110 ! JLS 7/89 V.2.0 !183
120 DIM A(1,300)!067
130 DEF LN(T$)=VAL(STR$(ASC(
T$)*256+ASC(SEG$(T$,2,1))))!
099
140 DEF CR$(S)=CHR$(INT(S/25
6))&CHR$(S-INT(S/256)*256)!
CONVERTS LINE NUMBER INTO CR
UNCH FORMAT !034
150 CALL BLUE !145
160 DISPLAY AT(1,11)ERASE AL
L:"RESEQ2" :: CALL CHAR(125,
"00FF"):: CALL HCHAR(2,13,12
5,6)!141
170 DISPLAY AT(3,5):"FANCY R
ESEQUENCING" !084
180 DISPLAY AT(5,1):"NAME OF
 FILE TO EDIT?":"DSK1.TEMP"
:: ACCEPT AT(6,4)SIZE(-12)VA
LIDATE(UALPHA,DIGIT,"_@."):S
$ !117
190 DISPLAY AT(7,1):"NAME OF
 NEW FILE?":"DSK1.NEWTEMP" :
: ACCEPT AT(8,4)SIZE(-12)VAL
IDATE(UALPHA,DIGIT,"_@."):D$
 !084
200 OPEN #1:"DSK"&S$,INPUT ,
DISPLAY ,VARIABLE 163 !203
210 X=0 !015
220 LINPUT #1:P$ :: X=X+1 ::
 A(1,X),A(0,X)=LN(P$)!072
230 IF A(0,X)>32768 THEN 270
 !085
240 DISPLAY AT(10,1):"READIN
G LINE #:";A(0,X)!135
250 IF SP>0 THEN 260 ELSE IF
 SEG$(P$,3,1)=CHR$(161)THEN
SP=A(0,X)!159
260 GOTO 220 !043
270 X=X-1 :: RESTORE #1 !055
280 DISPLAY AT(10,1):"FIRST
LINE # WAS: ";A(0,1)!229
290 DISPLAY AT(11,1):"FIRST
SUBPROGRAM: ";SP !193
300 DISPLAY AT(12,1):"LAST L
INE  # WAS: ";A(0,X)!017
310 DISPLAY AT(16,1)BEEP:"CH
ANGE FIRST LINE # TO #?" ::
ACCEPT AT(17,6)VALIDATE(DIGI
T)SIZE(5):Y$ !193
320 IF Y$="" THEN A(1,1)=100
 ELSE A(1,1)=VAL(Y$)!253
330 DISPLAY AT(16,1)BEEP:"US
E INCREMENT TO LINE #":"INC?
      TO?":"'ENTER' STOPS CH
ANGES HERE." !073
340 ACCEPT AT(17,6)VALIDATE(
DIGIT)SIZE(5):F$ !174
350 IF F$="" THEN IF L=0 THE
N DISPLAY AT(17,1):"":"" ::
GOTO 310 ELSE 410 ELSE F=VAL
(F$)!083
360 DISPLAY AT(18,1):"'ENTER
' TAKES CHANGES TO END" !174
370 ACCEPT AT(17,15)VALIDATE
(DIGIT)SIZE(5):T$ :: IF T$="
" THEN T=32767 ELSE T=VAL(T$
)!218
380 IF L>0 THEN 400 !147
390 A(1,0)=A(1,1)-F :: L=1 !
245
400 IF A(0,L)<=T THEN A(1,L)
=A(1,L-1)+F :: L=L+1 :: GOTO
 400 ELSE 330 !004
410 IF A(1,L-1)<=A(1,L)THEN
430 ELSE A(1,L)=A(1,L-1)+10
:: L=L+1 :: IF L=X+1 THEN 42
0 ELSE 410 !033
420 IF A(1,X)>32767 THEN 570
 !130
430 DISPLAY AT(20,1):"PRINT
LINE NUMBER TABLE TO?":"'NUL
L' TO NOT PRINT TABLE" :: AC
CEPT AT(22,1):DV$ :: IF DV$=
"" THEN 490 !092
440 IF L=X-1 THEN 430 !202
450 OPEN #6:DV$,OUTPUT !184
460 PRINT #6:S$,D$ !223
470 FOR Z=1 TO X :: PRINT #6
:A(0,Z);A(1,Z),:: NEXT Z !18
4
480 CLOSE #6 !156
490 OPEN #5:"DSK"&D$,OUTPUT,
DISPLAY ,VARIABLE 163 !037
500 FOR L=1 TO X :: SE=4 ::
LINPUT #1:P$ :: P$=CR$(A(1,L
))&SEG$(P$,3,LEN(P$)-2)!129
510 T=POS(P$,CHR$(201),SE)::
 IF T=0 THEN 550 ELSE TL=LN(
SEG$(P$,T+1,2)):: SE=T+3 !02
1
520 FOR LP=1 TO X :: IF A(0,
LP)=TL THEN TL=A(1,LP):: GOT
```

(See Page 16)

TRIALS OF A c99 BEGINNER

Polynomial curves

By CHARLES E. KIRKWOOD JR.

A program to evaluate a polynomial with integer coefficients was published in the March 1989 c99 article. This month will be an expansion of this program to calculate and plot as many as ten polynomials on the same graph.

Originally this program was written in FORTRAN and, as you know, the subscript values in FORTRAN start with 1 rather than 0. This program will ignore the zero subscripts completely and start with 1, which will make a slight modification to the March program.

Several new functions are written:

graph(nc,np,dx,dy,x,y,c,p,pr) where nc is the number of curves, np is the number of points, dx is the x increment or step, dy is the y increment or step, x is an array of the x values, y is a two-dimensional array of the y values for each curve, c is an array for the points on the graph for each curve, p and pr are used to designate whether the output is on the screen or printer. This function can print as many as 10 curves on the same graph. The function can be used with other programs to plot curves that are not polynomials.

y=ipoly(m,a,x) where m is the number of coefficients of the polynomial, a is an array of the coefficients, and x is the value of the independent variable. This function is a slight modification of the March function since the subscripts begin with 1 rather than 0.

r=divrnd(a,b) where a is the numerator and b is the denominator. This function is designed to round the integer division for plotting the curves. In this function, the result is rounded upward if the decimal fraction is more than one half and the result is positive and downward if the result is negative.

b=iabs(a) for the absolute value of a.

The program is more or less self-explanatory. Prompts are given for the input.

Generally not more than 4 or 5 curves can be seen well on the screen, but as many as 10 can be plotted on paper. If dy is input as 0, the graph function will calculate a dy value to make the graph fit on the screen or on a printer page. A total of 35 values of x can be plotted on the screen and 74 on the printer. The user may wish to design his own characters for the points; the letters of alphabet are used with this program, A for the first curve, B for the second, etc. The plotted point will be * when curves intersect.

Right now the output is formatted for 4 digits or a sign and 3 digits. This could be changed by the user.

```
/*polynomial evaluation with graph*/
#include DSK1.STDIO
extern atoi(),printf(),fprintf();
main()
{
  int aa[11],x[75],y[11][75],a[11][11],m[11];
  char c[11],s[10];
  int nc,n,mn,xo,xm,dx,dy,np,p;
  int i,j,k,pr,z;
  c[1]='A';
  c[2]='B';
  c[3]='C';
  c[4]='D';
  c[5]='E';
  c[6]='F';
  c[7]='G';
  c[8]='H';
  c[9]='I';
  c[10]='J';
  puts("Input 1 for screen output or\n");
  puts("      2 for printer output   ");
  p=atoi(gets(s));
  if(p==2)
    pr=fopen("PIO","w");
  puts("\nInput number of polynomials ");
  nc=atoi(gets(s));
  puts("\nInput degree and coefficients of polynomials\n");
  for(i=1;i<=nc;++i)
  {
    printf("\nDegree of #%d ",i);
    n=atoi(gets(s));
    mn=n+1;
```

(See Page 17)

EXTENDED BASIC—

(Continued from Page 15)

```
O 540 !055
530 NEXT LP :: GOSUB 580 ::
TL=32767 !000
540 P$=SEG$(P$,1,T)&CR$(TL)&
SEG$(P$,T+3,LEN(P$)-T-2):: G
OTO 510 !094
550 DISPLAY AT(24,1):"WRITIN
G NEW LINE #: ";LN(P$);:: PR
INT #5:P$ :: NEXT L !008
560 PRINT #5:CHR$(255)&CHR$(
255):: CLOSE #1 :: CLOSE #5
:: STOP !168
570 DISPLAY AT(1,1)ERASE ALL
:"BAD LINE #!" :: FOR J=1 TO
 X :: A(1,J)=A(0,J):: NEXT J
 :: GOTO 310 !135
580 FOR LP=1 TO X :: IF A(0,
LP)>TL THEN LP=X !167
590 NEXT LP !050
600 P$=SEG$(P$,1,LEN(P$)-1)&
CHR$(131)&" REFERENCE "&STR$
(A(1,LP-1))&" TO "&STR$(A(1,
LP))&CHR$(0):: RETURN !127
29505 SUB BLUE !149
29510 ! SWITCHES DISPLAY TO
WHITE ON BLUE; JLS 7/88 !230
29515 CALL SCREEN(5):: FOR L
=0 TO 14 :: CALL COLOR(L,16,
1):: NEXT L :: SUBEND !202
```

c99—

(Continued from Page 16)

```
  m[i]=mn;
  for(j=1;j<=mn;++j)
  {
    printf("\nCoefficient #%d ",j);
    a[i][j]=atoi(gets(s));
  }
  putchar(10);
}
puts("Input initial value of x ");
xo=atoi(gets(s));
puts("Input last value of x    ");
xm=atoi(gets(s));
puts("Input x increment        ");
dx=atoi(gets(s));
puts("Input y increment        ");
dy=atoi(gets(s));
k=1;
while(xo<=xm)
{
  for(i=1;i<=nc;++i)
  {
    mn=m[i];
    for(j=1;j<=mn;++j)
    aa[j]=a[i][j];
    y[i][k]=ipoly(mn,aa,xo);
  }
  x[k]=xo;
  xo=xo+dx;
  ++k;
}
np=k-1;
if(p==2)
{
  for(i=1;i<=np;++i)
  {
     fprintf(pr,"%4d ",x[i]);
     for(j=1;j<=nc;++j)
       fprintf(pr,"%4d ",y[j][i]);
     putc(10,pr);
  }
  putc(12,pr);
}
else
{
  putchar(12);
  for(i=1;i<=np;++i)
  {
    printf("%4d ",x[i]);
    for(j=1;j<=nc;++j)
      printf("%4d ",y[j][i]);
    putchar(10);
  }
  puts("Press any character ");
  z=getchar();
```

(See Page 18)

c99—

(Continued from Page 17)

```
  }
  putchar(12);
  graph(nc,np,dx,dy,x,y,c,p,pr);
  if(p==2)
  fclose(pr);
}

ipoly(m,a,x)
int m,x;
int a[];
{
  int i,y;
  y=a[1];
  for(i=2;i<=m;++i)
    y=y*x+a[i];
  return(y);
}

graph(n,m,dx,dy,x,yy,s,p,pr)
int m,n,dx,dy,pr,p;
int x[],yy[][75];
char s[];
{
  char pl[75];
  int y[11][75];
  int nn,nt,i,j,ymin,ymax;
  int line,iu,iv,val,iy,e;
  nn=0;
  nt=0;
  for(i=1;i<11;++i)
  {
    for(j=1;j<74;++j)
      y[i][j]=30000;
  }
  if(x[1]<=0)
  {
    nn=divrnd(x[1],dx);
    nt=1;
    nn=iabs(nn)+1;
  }
  ymax=yy[1][1];
  ymin=ymax;
  for(i=1;i<=n;++i)
  {
    for(j=1;j<=m;++j)
    {
      if(yy[i][j]<ymin)
        ymin=yy[i][j];
      else if(yy[i][j]>ymax)
        ymax=yy[i][j];
    }
  }
  if(dy==0)
  {
    if(p==2)
      dy=(ymax-ymin)/60+1;
    else
      dy=(ymax-ymin)/21+1;
  }
  iu=divrnd(ymin,dy)-2;
  for(i=1;i<=n;++i)
  {
    for(j=1;j<=m;++j)
      y[i][j]=divrnd(yy[i][j],dy);
  }
  iv=divrnd(ymax,dy)+1;
  while(iv!=iu)
  {
    val=iv*dy;
    for(i=1;i<74;++i)
      pl[i]=' ';
      if(nt==1)
        pl[nn]='|';
      if(iv==0)
      {
        for(i=1;i<=m;++i)
          pl[i]='-';
        if(nn>0)
        {
          i=nn;
          while(i<=m)
          {
            pl[i]='+';
            i=i+10;
          }
        }
        i=1;
        while(i<=nn)
        {
          j=nn+1-i;
          pl[j]='+';
          i=i+10;
        }
      }
      for(i=1;i<=n;++i)
      {
        for(j=1;j<=m;++j)
        {
          iy=y[i][j];
          if(iv==iy)
          {
            if((pl[j]>='A')&(pl[j]<='J'))
              pl[j]='*';
            else
              pl[j]=s[i];
          }
        }
      iv=iv-1;
      if(p==2)
        fprintf(pr,"\n%4d ",val);
      else
        printf("\n%4d ",val);
      for(i=1;i<=m;++i)
      {
        if(p==2)
          putc(pl[i],pr);
        else
          putchar(pl[i]);
      }
    }
    dy=0;
    return;
}

divrnd(a,b)
int a,b;
{
  int c,d,e;
  c=a/b;
  d=a%b;
  e=b/2;
  d=iabs(d);
  e=iabs(e);
  if(d>e)
  {
    if(c<0)
      c=c-1;
    else
      c=c+1;
  }
  return(c);
}

iabs(n)
int n;
{
  int m;
  if(n<0)
    m=-n;
  else
    m=n;
  return(m);
}
```

THE MAKING OF A PORTABLE TI

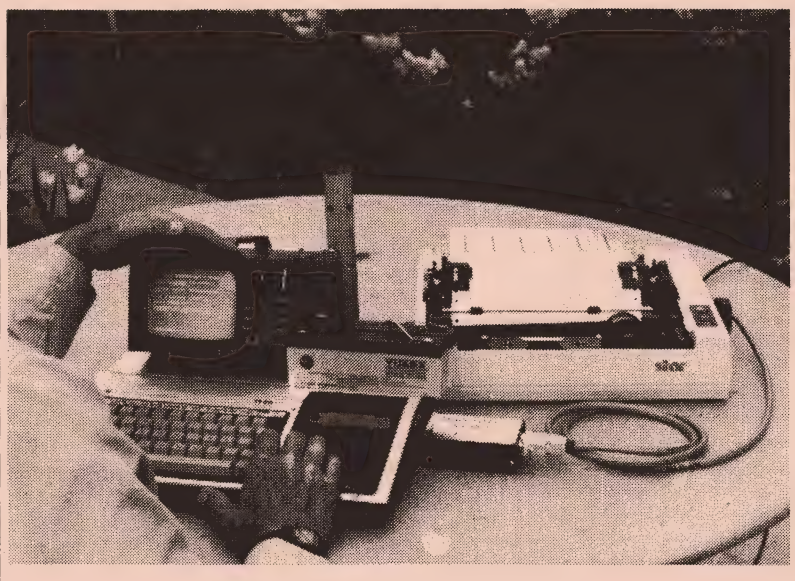
Making room for the RAMdisk means condensing 32K memory circuitry

By JAN JANOWSKI

This is the second in a three-part series about building a TI console that includes a memory expansion, RAMdisk and printer port.—Ed

It was quickly beginning to dawn on me that the size of this project was larger than that of my first portable. I needed the inclusion of a 32K memory expansion on this card, and it had to take up as little space as possible. The 32K memory had to be built and tested *before* I could build the RAMdisk, but the RAMdisk had to be laid out around the 32K memory mod!

I solved this classic problem, of "which came first, the chicken or the egg," by making a photostat of the expansion board and making paper cutout models of the integrated circuits. I then placed them on the copy page and moved them around until the layout design looked about right. But even then I still had a problem in that the memory expansion was taking up too much space.



The portable TI in full operation

CONDENSING THE CIRCUITRY

I gave up trying to create a layout design for both and started using a "proto-board," and I worked on the 32K memory mod exclusively, focusing my efforts on getting the size of the circuitry as small as possible. I wonder how many of you are aware of or have bought the TI hardware modification manual that the Chicago Users Group is selling (the cost of the manual, with postage and handling, is $10.)? Let me tell you, if you are into hardware mods, it is worth its weight in gold. It includes a schematic and board layout for a 32K memory upgrade, using just one chip, and I used that design as a basis from which to start.

This 32K memory expansion module can be built using only one chip, one transistor, eight diodes, and four resistors, and the size ends up being 2.75 inches by 1.75 inches. I later ended up eliminating the transistor and one resistor; the size of the board space that I am now using is 1.5 inches by .75 inches (only 25 percent as much space). Compare this 1.24 square inches to a 32K card for an expansion box.

Parts were eliminated by picking up one more signal from the computer board, ("Not DBIN" from U508 pin 9), rather than using "DBIN" from the GROM socket. By doing this, I eliminated the need for the transistor and pullup resistor. Also, by moving the diodes around carefully, I was able to install all the circuitry *underneath* the IC socket.

Using this reduced sizing for the 32K memory upgrade, it was possible to layout the RAMdisk, and with the RAMdisk and 32K laid out, there was still 26 square inches of board space available.

WIRING THE BOARD

I next started with the wiring on the expansion board. I had earlier decided that if I didn't develop some type of wiring scheme, I would go crazy trying to find a broken wire once things were completed. Can you imagine hundreds of wires all of the same color tightly wired on a board, and one end of one wire comes off?

Therefore, I decided on a color code to at least reduce my level of confusion a wee bit. For ground, I used black wire. For +5v, I used red wire. For the data bus wiring, I used blue wire. For the address bus, I used yellow wire. And for wiring not of the above types, I used white wire. This one bit of forethought saved me untold hours of frustration. (See parts layout and board size drawing.)

As I started transcribing wires from schematic to expansion board, I uncovered one signal that occurs in the expansion box that does not occur inside the console. You will not find the equivalent of pin 11, of the PEB board, "Not RDBEN," in the console, because this signal is unique to the PEB. What this signal does is as follows:

Let's say that a RAMdisk inside the expansion box wants to send some data to the computer. The RAMdisk puts a "low" on pin 11, which disables any other card in the PEB from outputting data, and simultaneously enables the output data tranceiver on its own board. Here is what I mean by "enabling the data tranceiver." The data bus, unlike the address bus, sends data in both directions, to and from the computer. Tranceivers both send and receive, but these functions cannot be done *simultaneously*. Picture a traffic cop keeping the two-way traffic from colliding on a one lane street — this is exactly what happens. "DBIN" enables the input side (data

(See Page 21)

PORTABLE TI—

(Continued from Page 20)

from computer), while the logic that creates "Not RDBEN" enables the output side (data *to* computer).

My problem was that "Not RDBEN" does not exist in the console, and though the RAMdisk might work inside the console, would it work in conjunction with the PEB?

Fortunately, it turns out that it does, and the "Not RDBEN" signal can be disregarded in the console, for this project, at least.

Remember pin 11? Let's follow it to the console. Pin 11, "Not RDBEN," from any PEB card, goes to the PEB expansion card (the interface card, which usually sits in slot one of a PEB) where it is buffered, renamed "Not RDBENA.A" (which does the exact same thing it did before) and is passed directly to the black cable and to the plug at the right side of the TI console. Inside this black connector, there are more electronics, and this signal "Not RDBENA.A" becomes the enable for the data bus tranceiver in the large black plug in the side of the TI console. Then, and only then, will data get inside the console from the PEB.

You can imagine my concern that the two separate data bus enables would cause data collisions, but, thank goodness, they haven't.

So, I continued on, and on with the wiring — I stopped only after the ground bus wiring was installed. Then I put it all together to verify it wouldn't crash the computer. I then took it apart, added the +5v wiring, reassembled it, and tested it again.

I took it apart again and added the 32K memory expansion, put it back together, and tested it again. This went on, and on. I finally got to the point where the 32K memory was functional. (I tested its functionality with the TI Diagnostics program and a Mini-Memory Cartridge.) Once I got that far, I was able to use this 32K memory to run other tests.

One thing led to another. Now I was finally beginning to see some results from my work. You can see the placement of the parts on the expansion board and the cutouts and notches needed to have the ex-

(See Page 26)

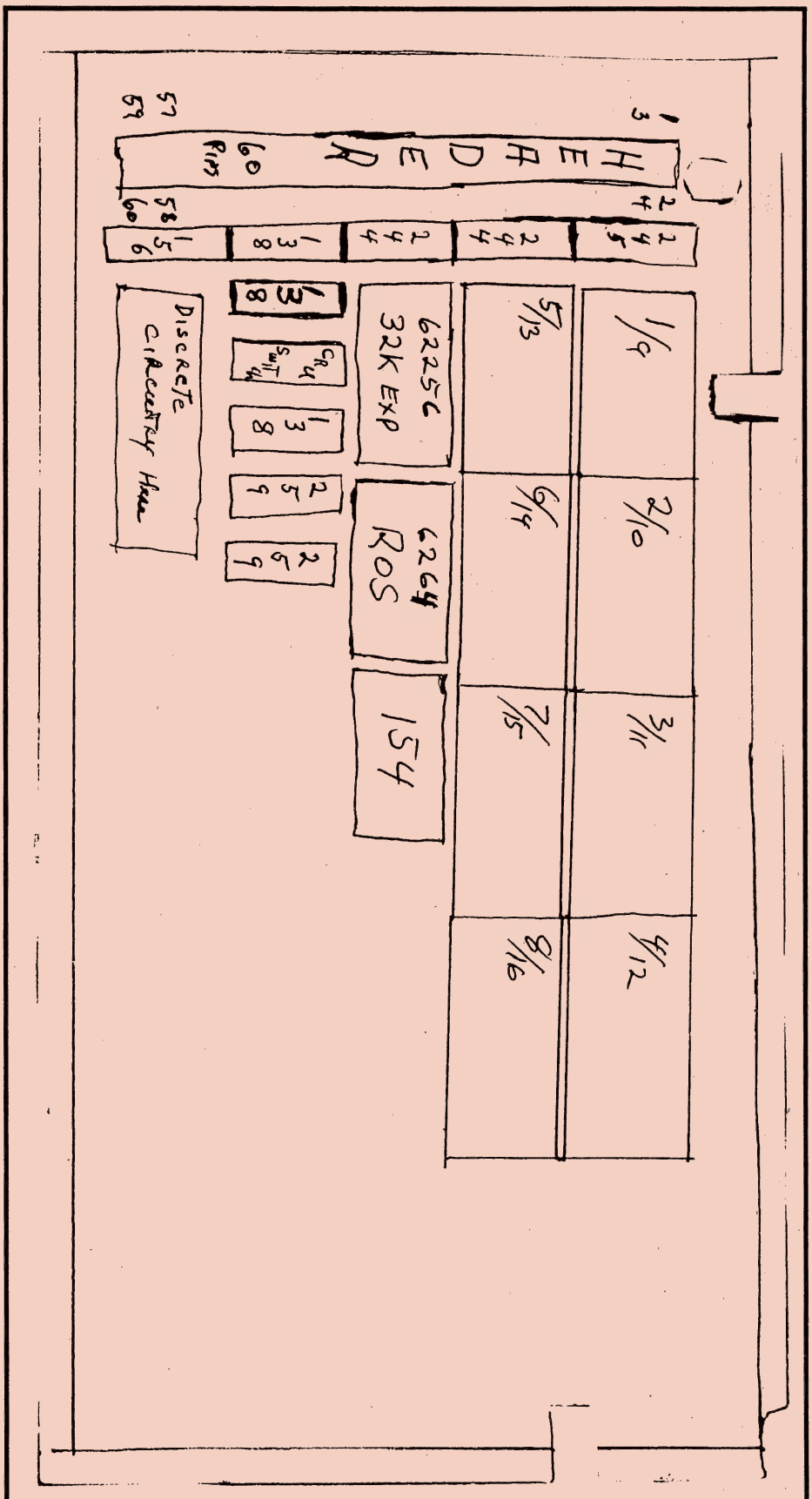


Fig. 1: Parts Layout. Note cutouts and notches and the amount of free space. (Layout reduced 90% from original)

PORTABLE TI—

(Continued from Page 21)

pansion board fit in the top cover. (See Fig. 1)

HAMSOFT MODULE

At this point, the expansion board contains the 32K RAM, and most of a RAMdisk, with areas of the board set aside for the parallel port, which is the next hurdle to get over. My only concern is that, although the Hamsoft Module's parallel port works with a RAMdisk in the console, I don't know whether it will work in conjunction with an expansion box. Unfortunately, I did not have a connector that would allow me to plug in the Hamsoft as well as the expansion box for testing. That was something that would just have to wait until later.

Some important thanks goes to Kantronics (1202 E 23rd St., Lawrence, KS 66046) for making the Hamsoft Module for the TI computer, as without it I would not have attempted this project in the first place. The Hamsoft Module allows the interfacing of a stand alone TI computer to a Ham radio, for RTTY and CW operation (Radio Teletype and Morse Code), for both reception and transmission. With my TI, I have enjoyed many years of communications, worldwide, with other Hams on RTTY. Therefore, I have a special feeling for the harmonious combination of my TI and my Ham radio rig. The Hamsoft Module also contains a parallel port, and it was this parallel port that deeply interested me, for use in the portable.

BACK AT THE RAMDISK

Meanwhile, back at the RAMdisk, I was getting near the end of the wire installation, and was glad of that. As mentioned, after each major wiring session with the expansion board, it was tested to verify that the console was not being locked up by an error on this board. It takes longer to do things that way, but you have a better idea of where an error is this way. Finally the day came when I could stuff the sockets with logic chips and turn it on for a test. The console did not lock up, but other than the 32K mod, which had already been tested, it was not working. I eventually found three errors, and once they were fixed, it worked. Eureka! (See Fig. 2)

The schematic I used for the RAMdisk is a modified version of a standard RAM

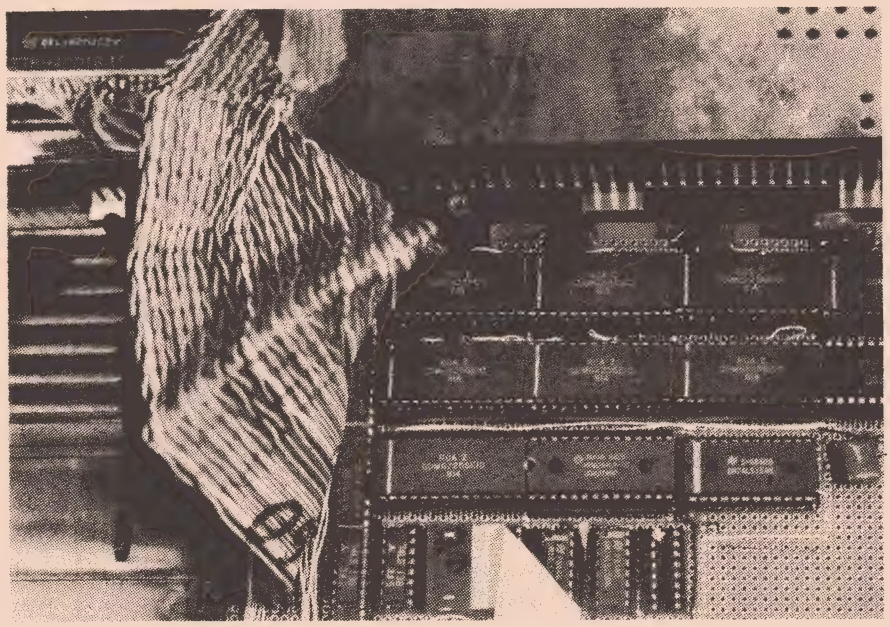


Fig. 2: Expansion board showing cutout in board for support.

disk. It was scaled down in that it uses only 16 62256 chips instead of 32. The total size of the RAMdisk is 512K. The second 74LS154 and its decoder, 74LS259 were eliminated. I added a 1K pullup resistor between the memory chip pins 28 and 27, so that, during power off, it would not inadvertently try to write to memory. I also wanted a very stable battery backup so a memory capacitor was added (Radio Shack 272-1440). This worked so well that I placed one in the "supercart," too.

There is no LED on the modified power supply, and I wanted to do something special, so I re-designed the LED for the RAMdisk, so that that one LED had two different colors in it. The red part is connected to the power supply, and glows dimly red when the power is on. The green side is connected to the RAMdisk, and glows bright yellow when the RAMdisk is accessed. This LED is placed beside the "4A" stamped in the top chrome trim of the console.

CRU SELECTION SWITCH

During my tests I found it desirable to be able to change CRU addresses, so I mounted the CRU selection switch under the keyboard, added access to it from the bottom of console. A short ribbon cable and 16-pin header plug connects the remote switch to the expansion board IC socket. I chose to use 4 AA nicad batteries

(See Page 27)

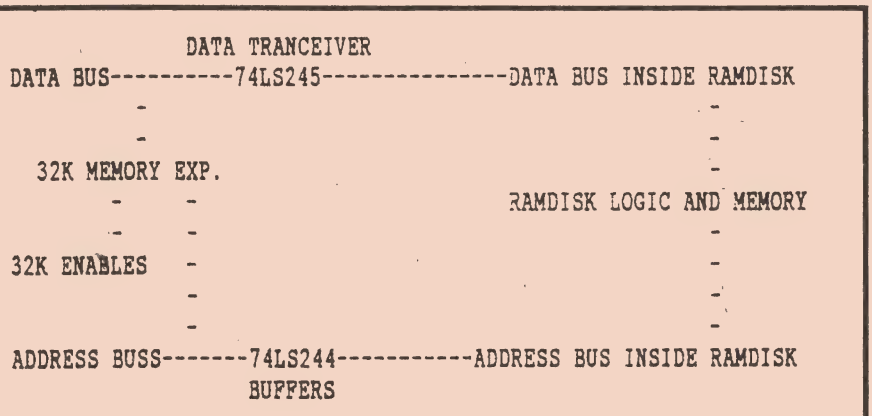


Fig. 4: Interconnects within the console. (Console attaches to left side)

PORTABLE TI—

(Continued from Page 26)

to provide for longer battery backup of the static RAM chips. The placement of the batteries has moved a few times, and they are presently mounted under the keyboard. All this extra circuitry inside the console increases the weight significatly, but from the outside, the only hint that this is not a stock console is that dual-colored LED beside the "4A." See Fig. 3.

To give you an idea of the flow of the interconnects within the console, Fig. 4 is a block diagram of the additional circuits, and their placement in the flow of things. The TI console attaches to the left side of the drawing.

Well, am I proud of myself! I managed to squeze a RAMdisk inside a console, the combination console/RAMdisk will work on AC or DC, and there is plenty of room left over for the parallel port which, in preliminary tests, seemed to work well. And, let's not forget the 8-bank supercart. Talk about being the happiest guy in the TI world!

In test after test, including the TI diagnostics, Memtest, and Megtest, all tests indicate that there is no problem. After I loaded the version 7.3 RAMdisk Operating System (ROS) and John Johnson's RAMdisk Menu program, I started loading the RAMdisk's memory storage space. It was then, when I was about to announce to the world that I was successful, that disaster struck — I disconnected the console from the expansion box, and on power-up the Menu program ran. I selected Funnelweb, and the loader part worked, but it crashed when I attempted to load the text editor. I reset the computer and selected TI-Artist, which promptly crashed. Next I selected John Johnson's Remind-Me!, and it loaded, but I was unable to load the reminders for the month from the RAMdisk.

I then tried a simple BASIC program which loaded with the standard RUN "DSK5.FILENAME" command, and that worked. Then, using the Editor/Assembler cartridge, I was able to load the text editor from Funnelweb, but after a document was

(See Page 28)

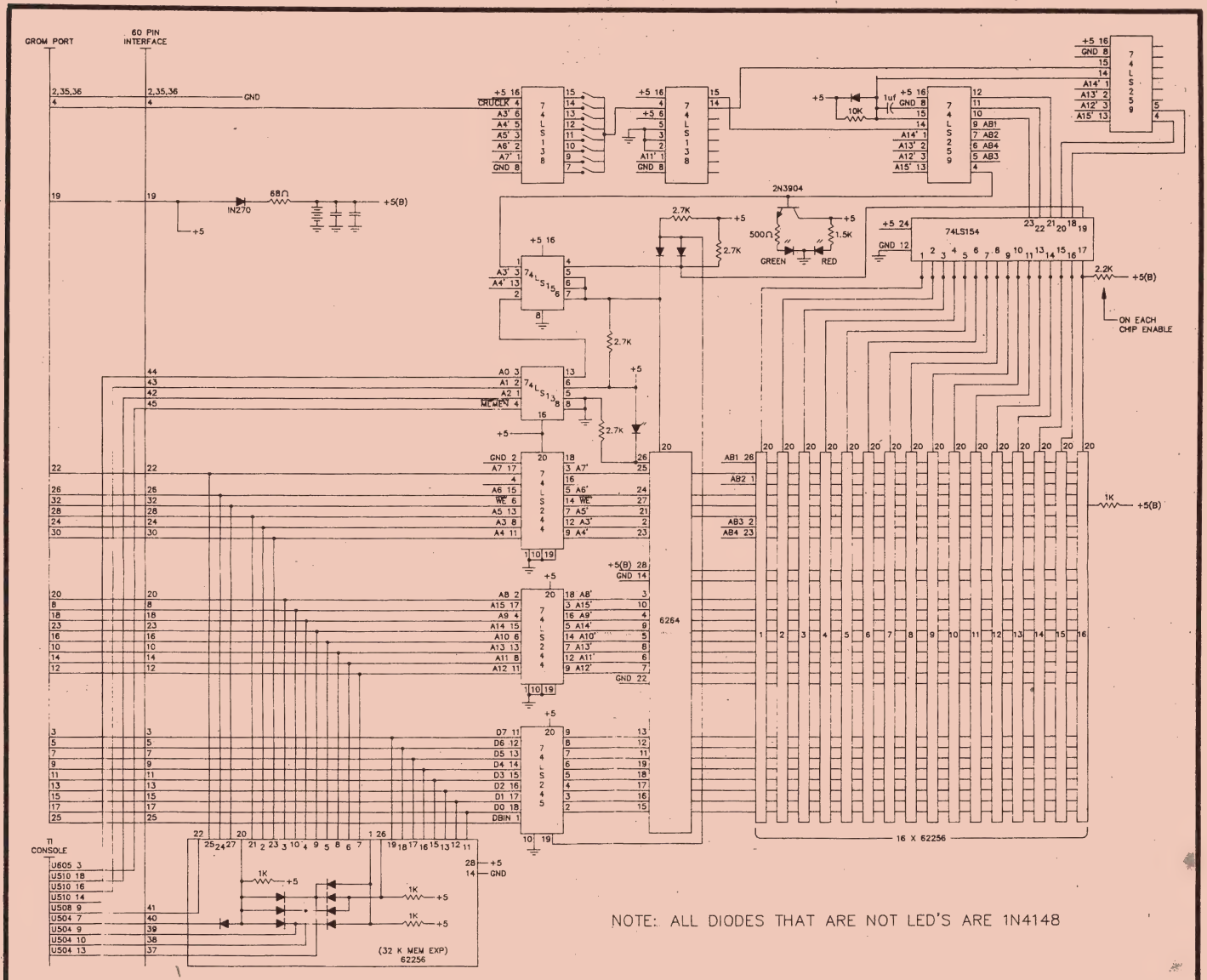


Fig. 3: Schematic of RAMdisk drawn by Matt Babich

PORTABLE TI—

(Continued from Page 27)

completed, I was unable to store it or, for that matter, load a text file from the RAMdisk.

Next, I tried Fast-Term, and that program seemed to run fine, but I was unable to fully test it — the portable did not have a serial port to test it on. A few more hours of testing did not render any consistent results except that whenever I connected the expansion box to the console, there was absolutely no problem. However, whenever I removed the expansion box, the strange problems reappeared. I couldn't help asking myself the question: What's going on here?

When I removed the disk controller card (dcc), with the PEB attached, the problems reappeared! I next tried leaving the dcc inside the PEB, but I disconnected the disk drives. I obviously had no disk access, but I didn't have the problem of crashing either; with this arrangement, the system seemed happy. I then thought back to the day I was at another user's house, and he showed me that he could load a program without a disk controller card. I called him, and he tested it for me again, this time more throughly. Now, when he ran the programs, he encountered the same problems I had; some programs ran fine, however. Strange!

I then tried to fool the computer by changing the CRU address of the RAM disk to 1100, which is the standard disk controller CRU address (after separating the console from expansion box). That did not help.

What followed was 4-5 days of crazy tests, all of which were so inconclusive that I will not mention them here. This occurred about two weeks before the 1988 Chicago TI Faire, so I went to the Faire armed with schematics, drawings and as much documentation as possible, and talked with anyone who would listen.

IT'S IMPOSSIBLE

At the Faire, my idea and my experimentation were met with various responses; though the words differed, the meaning was the same: "IT'S IMPOSSIBLE—IT CAN'T BE DONE." In more than a few instances, had I had not had my schematics and drawings, I seriously believe that I would have been dismissed as a crackpot.

I was not banking on the Faire alone as a source of answers to my problem. I sent off letters to virtually everyone who seemed to matter in the TI community, including Texas Instruments. In each letter, I asked whether I was in fact correct about the lack of a dcc causing the kind of problem that I was experiencing. I finally got a call from Ron Walters, in Ohio, who talked with me at length about my project, and that conversation was followed up with a 5-minute face-to-face conversation with Bud Mills, of Bud Mills Services, at the December Chicago Users' Group meeting. In our discussion, of this matter, Bud promised that he would do all that he could find a fix for this problem.

Well, exactly what was my problem? Here is my understanding of the problem that I was facing: The ROS is written to co-exist with the disk controller's Device Service Routine (DSR). When a disk system is installed in an expansion box, you lose 2K of RAM in the VDP area, and this is set aside for the DSR routine for handling the disk accesses. Apparently, the RAMdisk ROS works in conjunction with this disk DSR, and, by removing the disk controller, the ROS was attempting to communicate with a DSR that wasn't present. Therefore, it crashed.

I later received an answer, of sorts, from Texas Instruments, which I have copied here for your reading pleasure.

"1. Install enough of the disk controller ROM (and supporting hardware) on board the computer to simulate the full capabilities for the disk controller in setting up the PAB." (Here I must admit that I do not know how much is enough.)

"2. Contact the author of the ROS to understand what parts of the disk controller ROM software it uses. Then, modify the ROS software to incorporate the missing segment. Mr. Janowski, I hope these suggestions are helpful in completing your project. The research (building sample circuits and disassembling ROM software) would be very extensive. We have reached the end of our expertise and will not be able to assist you any further in this matter."

As a result of this communication, I immediately started looking for this disk drive DSR in memory and found it with the Mini-Mem cartridge. In order to make an effective search of the machine's memory, I powered up the system and browsed the VDP memory, documenting what I found. I next powered down the system and removed the disk controller card. After returning power to the system, I browsed the VDP RAM again to see what was missing. Here are the differences that I found:

37D8	AA	3EEB	05	3FF5	05
37D9	3F			3FF6	4D
37DA	FF			3FF7	45
37DB	11			3FF8	4E
37DC	03			3FF9	55
37DD	02			3FFA	20
37DE	04			3FFB	20
37DF	50			3FFC	20
37E0	00			3FFD	20
37E1	04			3FFE	20
				3FFF	20

All other locations were 00, regardless of whether the dcc was installed or not. (The card, by the way, was a Myarc DSDD disk controller.)

The third and final part of this series will be published next month.—Ed

Your Computer Deserves A Great Monitor

TEX★COMP™ Proudly Introduces The MAGNAVOX 8CM515 RGB/Composite Color Monitor 80

ANALOG RGB MODE FOR THE MYARC 9640

17% Larger Screen than Standard 12" Monitors • 14" tube provides larger screen size without an appreciably larger footprint.

RGB TTL, RGB Analog, Composite Video Inputs • make monitor compatible with all three standard video signal types.

Green Raster Display Switch • emulating monochrome performance for text applications, this switch disables the monitor's ability to display anything other than green light — a feature normally found on more expensive products.

Etched Faceplate • the CRT face is treated with a special glare-reducing process to help eliminate eye fatigue caused by reflections from the monitor face.

Dark Glass CRT • the tube glass used in this product incorporates a special dark background material which helps to improve image contrast and reduce operator eye strain — a feature normally found on more expensive products.

640 Dots of Resolution (Horizontal) • in RGB, the monitor system will display 640 dots of image resolution on each scan line, the minimum number required to display 80 characters.

240 Lines of Resolution (Vertical) • the monitor will produce 240 vertical scan lines, standard for RGB use.

Displays 2000 Characters, 80×25 • this monitor system will display 25 lines of 80 characters each.

Comb Filter on Composite Input • when used with a composite input signal, the monitor's comb filter improves the resolution quality of the visual image.

Line Level Audio Input • provides convenient connection to a VCR or any device which generates a line level audio signal.

Image Size and Position Controls • located on the back panel, these controls allow the user to adjust both the size and position of any image displayed on the CRT, assuring compatibility with computer and other video devices.

Built-in Tilt Stand • allows the user to adjust the viewing angle to suit personal preference.

Two Year Parts and Labor Warranty • solid evidence that this is a reliable product, nationally supported by Magnavox.

THE MAGNAVOX 14" 8CM515 COLOR MONITOR IS BY THE FINEST COLOR MONITOR EVER AVAILABLE FOR THE TI-99/4A AND THE ONLY MONITOR THAT IS FULLY COMPATIBLE WITH EVERY TI-99/4A RELATED MODEL AVAILABLE INCLUDING THE BASIC 99/4A, THE MYARC GENEVE, THE TRITON TURBO, THE DIJIT & MEGATRONICS 80 COLUMN CARDS, NOT TO MENTION ALL OTHER AVAILABLE COMPUTERS. THE DISPLAY WITH THE TI-99/4A IS DYNAMITE AND WITH THE DIJIT 80 COLUMN CARD IT BOUGHT THE HOUSE DOWN AT THE TI-99/4A FEST WEST '89. ALSO HAS AN AUDIO OUTPUT FAR SUPERIOR TO WHAT HAS BEEN AVAILABLE. THIS MONITOR RETAILS FOR $599 BUT TEX-COMP HAS PURCHASED AN ENTIRE TRUCK LOAD TO BRING YOUR PRICE DOWN TO $279.95 AND WE EVEN THROW IN A HEAVY DUTY TI-99/4A MONITOR CABLE, A $14.95 VALUE. COMES WITH A 2 YEAR NATIONAL FACTORY WARRANTY FROM MAGNAVOX (NORTH AMERICAN PHILLIPS) WHICH IS THE LONGEST IN THE INDUSTRY.

$279.95+$10S&H

TEX★COMP™

AMERICA'S NUMBER ONE TI COMPUTER RETAILER

Charge-it On Your Visa or MasterCard
ORDER BY PHONE
24 HOURS A DAY
7 Days a Week!

(818) 366-6631

P.O. Box 33084, Granada Hills, CA 91344

TERMS: All prices F.O.B. Los Angeles. For fastest service send cashiers check or money order. Personal checks require up to 15 days to clear. Prices reflect a 3% discount for cash or approved check. Add 3% for Credit Card orders. Prices and availability are subject to change without notice. We reserve the right to limit quantities. California orders add 6.5% sales tax.

LOADERS, MODULAR PROGRAMMING, LINKAGES & OVERLAYS

Overlays using Extended BASIC

This is the fifth and final installment of a series on loaders, linkages and overlays.—Ed.

By MERLE VOGT

Overlays with Extended BASIC are much different from those described last month. There is no UTLTAB to keep track of addresses as was used by Mini-Memory or the Editor/Assembler.

Extended BASIC keeps high RAM, >A000 and up, for itself. The primitive initialization pointers are set to use low RAM space to contain any loaded modules.

The resulting address table contains only two entries: FSTLOW and LSTLOW.

FSTLOW at >2002 contains >24F4

LSTLOW at >2004 contains >4000

The Extended BASIC loader observes the following procedure when a CALL LOAD command is executed:

A. It places the program code into low RAM starting at FSTLOW, >24F4.

B. It updates FSTLOW to reflect length of the module.

C. It subtracts 8 from LSTLOW, at >2004, getting address >3FF8.

D. It places the module name, and entry address in the space >3FF8 through >3FFF. Thus the Extended BASIC loader creates REF/DEF table entries downwards, as we are accustomed to seeing it done. Note that the Extended BASIC loader will not process any references, REFs. You cannot use them in any assembly modules to be used with Extended BASIC.

Considering the above addressess, we can see that allowing space for some REF/DEF table entries, we have about >3FE0->24F4 = >1AFC, or 6808 bytes of space in low RAM for modules. So we cannot load many, or very large, modules using Extended BASIC as it is fundamentally structured.

The use of high RAM by Extended BASIC is related to how much area is demanded by data, arrays, strings, etc. in running the program. By keeping the Extended BASIC program minimal, using it strictly as a driver and loaded of assembly modules we then can use a large portion of high RAM for overlay modules. I have experimented and believe that the area >A000 through >F000 generally can be used without conflicts.

We can create and assemble two types of modules: relocatable and absolute origin. With Extended BASIC, both are most feasible. The Extended BASIC loader normally expects to be fed relocatable modules, which it loads into low RAM, for which it updates FSTLOW, LSTLOW and the REF/DEF table. Where we must run programs that exceed low RAM space, absolute origin modules in high RAM are required.

The Extended BASIC loader can handle absolute origin modules, but the whole procedure changes. It does only two things:

A. It loads the module into RAM at the address on the AORG statement in the module.

B. It places the name, and address, of the module into the REF/DEF table and updates LSTLOW to reflect it.

Note that you can "AORG" into low RAM for modules, but it demands strict program organization.

Let us review:

A. You cannot use any REF to a DEF in any other module.

B. Instead the EQU (EQUate) must be used.

C. It will blindly overlay modules on top of module you may still be using if you get careless.

D. The AORG addresses, timing of CALL LOAD commands, length of modules, etc. must be strictly controlled.

E. You must erase names of used modules from the REF/DEF table before loading more modules.

F. You must update the LSTLOW address to reflect the cleared REF/DEF table space.

G. You must not erase any REF/DEF table entry still needed.

All the above makes it appear that using overlay procedure is excessively complex. Yes and no. Complex, yes. But it is powerful and flexible. You can run programs of any size. You can reload and rerun modules as needed. The *cleanup* steps discussed above can be put in a small module and called as each overlay is needed. Then each overlay module can exit into the cleanup module. It, in turn, exits back to Extended BASIC which loads the next step in the chain. The cleanup module is loaded first and its entry in the REF/DEF table must not be erased.

Alternately, you can use a routine in the Extended BASIC code to do the cleanup, as outlined last month. You would execute the procedure before each load of a new module. Note that the loader manages the REF/DEF table as a *stack*, in a primitive way. The cleanup routine has to *unstack* the table so that LSTLOW points to space for the new entries.

In the following paragraphs, I will set up a program system which copies a large part of the one shown last month.

A. Routes AAAAXX, BBBBXX and CCCCXX are similar to AAAAMM from last month, except that EQUates replace all REFs. We will have to add our own standalone scroll routine. Extended BASIC does not have a GPLLNK function. This module must be loaded first and never overlayed. (See program SCROLL.)

B. For educational purposes, I will delete the routine 6000 from last month, which resets pointers, and give you an assembly module. It is coded to fit Extended BASIC requirements. It also must be loaded and

(See Page 31)

LOADERS, MODULAR PROGRAMMING, LINKAGES & OVERLAYS

(Continued from Page 30)

never overlayed.

C. The Extended BASIC MASTER control program is similar to the MASTER program used last month. Some CALL LOAD commands are added and routine 6000 is deleted.

Here is a summary of the program structure:

A. SCROLL module. Loaded and never disturbed.

B. RESET module. Loaded and never disturbed.

C. Modules AAAAXX, BBBBXX and CCCCXX are all overlays.

D. Extended BASIC MASTER control program.

SCROLL is an example of how to make a standalone scrolling routine in assembly. You can use this in the next 500 program systems you write. The action of scrolling is nothing but moving each line up one line and erasing line 24 to blanks. Use VMBR to read a line, then VMBW to write it one line higher on the screen, and loop 23 times. Then use VMBW to blank out line 24. (See Fig. 1.)

Module RESET resets the pointers to make the loader put modules AAAAXX, BBBBXX and CCCCXX into low RAM at address >24F4. This pointer (FSTLOW) in in RAM >2002. It must be reset to the value >24F4.

Then the REF/DEF table start address pointer, LSTLOW, at address >2004, must be reset to the value >3FF0. We must reset it to this value to avoid destroying the DEF entries for SCROLL and RESET. Then the DEF table entry for the overlay modules must be erased each pass. (See Fig. 2).

The Extended BASIC Master Control Program is coded somewhat like the control program in last month's installment. It must load modules SCROLL and RESET before starting the overlay steps.

```
100 REM RUNNING MULTIPLE OVE
RLAYS IN X-BASIC ENVIRONMENT
 !047
110 CALL INIT !157
120 NEXTRTN=1 !235
130 CALL LOAD("DSK1.PT5/SCLO
BJ")!210
140 CALL LOAD("DSK1.PT5/RESO
```

(See Page 32)

```
S1          AORG >A000
S2          DEF  SCROLL
S3  SCROLL  BLWP @SCRL
S4          B    *R11
S5  VMBR    EQU  >202C
S6  VMBW    EQU  >2024
S7  WS      BSS  32
S8  LINE    BSS  32
S9  CLER    TEXT '
S10 SCRL    DATA WS,DOIT
S11 STAT    EQU  >837C

S12 DOIT    LI   R3,>20
S13         LI   R4,0
S14         LI   R5,23

S15 LOP23   MOV  R3,R0
S16         LI   R1,LINE
S17         LI   R2,32
S18         BLWP @VMBR
S19         MOV  R4,R0
S20         LI   R1,LINE
S21         LI   R2,32
S22         BLWP @VMBW
S23         AI   R3,32
S24         AI   R4,32
S25         DEC  R5
S26         JNE  LOP23

S27         MOV  R4,R0
S28         LI   R1,CLER
S29         LI   R2,32
S30         BLWP @VMBW
S31         CLR  @STAT
S32         RTWP
S33         END
```

Program explanation (SCROLL)

Line No.	Explanation
S1	Set origin to AORG >A000 in high RAM
S2	DEFines the module name
S3-S4	Start the run
S5, S6, S11	Set up needed EQUates
S7-S10	Set up data and text items. (Line S9 has 32 spaces)
S12	Pointer to screen line 2
S13	Pointer to screen line 1
S14	Loop counter value 23
S15-S26	Scroll up one line, increment counters R3, R4 and test for finish
S27-S30	Blank line 24 of screen
S31-S32	Clear status and exit

Fig. 1 (Line numbers are for reference only, do not enter them)

LOADERS, MODULAR PROGRAMMING, LINKAGES & OVERLAYS

(Continued from Page 31)

```
BJ")!218
150 ON NEXTRTN GOTO 1000,200
0,3000,4000 !197
160 REM!154
1000 REM RESTORE THE POINTER
S !115
1010 CALL LINK("RESET")!083
1020 REM GET MODULE AAAAXX.
!130
1030 CALL LOAD("DSK1.PT5/AAX
OBJ")!202
1040 REM NOW RUN IT. !174
1050 CALL LINK("AAAAXX")!133
1060 NEXTRTN=3 !237
1070 GOTO 150 !229
2000 REM RESTORE THE POINTER
S !115
2010 CALL LINK("RESET")!083
2020 REM GET MODULE BBBBXX !
088
2030 CALL LOAD("DSK1.PT5/BBX
OBJ")!204
2040 REM RUN IT !108
2050 CALL LINK("BBBBXX")!137
2060 NEXTRTN=4 !238
2070 GOTO 150 !229
3000 REM RESTORE THE POINTER
S !115
3010 CALL LINK("RESET")!083
3020 REM GET MODULE CCCCXX !
092
3030 CALL LOAD("DSK1.PT5/CCX
OBJ")!206
3040 REM RUN IT !108
3050 CALL LINK("CCCCXX")!141
3060 NEXTRTN=2 !236
3070 GOTO 150 !229
4000 REM END OF JOB !065
4010 DISPLAY "END OF OVERLAY
 JOB": : :!099
4020 STOP !152
```

Here I will let you complete routine 3000. It refers to module CCCCXX and DSK1.PT5/CXOBJ. NEXTRTN equals 2.

Routine 4000 is the same as in last month's master control program (Page 17, June 1989).

Module AAAAXX (Fig. 3) is different in several areas from the code shown in AAAAMM (Page 16, June 1989). Note that the AORG should not be used in this module. So the Extended BASIC loader will load the module into low RAM at the default address of >24F4.

```
R101            AORG    >A200
R102            DEF     RESET
R103    RESET   BLWP    @RSET
R104            B       *R11

R105    WS      BSS     32
R106    RT2002  DATA    >24F4
R107    RT2004  DATA    >3FF0
R108    STAT    EQU     >837C

R109    RSET    DATA    WS,RST
R110    RST     MOV     @RT2002,@>2002
R111            MOV     @RT2004,@>2004
R112            LI      R3,8
R113            LI      R4,>3FE8
R114    RLOP    CLR     *R4+
R115            DECT    R3
R116            JNE     RLOP

R117            CLR     @STAT
R118            RTWP
R119            END
```

Program explanation (RESET)

Line No.	Explanation
R101	Absolute origin to >A200, past module SCROLL
R102	DEFine name RESET
R103-R104	Start execution
R105-R107	Workspace and values for reset ponters. Note value in item RT204. We cannot reset location >2004 to >4000 because that would destroy the REF/DEF table entries for SCROLL and RESET modules.
R108-R109	BLWP vectors
R110-R111	Reset pointers for FSTLOW and LSTLOW
R112-R116	Clears one entry from REF/DEF table. These would be for AAAAXX, BBBBXX and CCCCXX
R117-R118	Exit module

Fig. 2

Module BBBBXX can be made from the code of AAAAXX by changing four lines, and saving the result with the new name.

```
1           DEF     BBBBXX
2  BBBBXX   BLWP    @AA2
7  MA       TEXT    'MESSAGE FROM
                    BBBBXX '
8  MI       TEXT    'PASSING CONTROL
                    TO EOJEOJ'
```

Similarly, module CCCCXX can be made by changing the same four lines again, and saving that result.

```
1           DEF     CCCCXX
2  CCCCXX   BLWP    @AA2
7  MA       TEXT    'MESSAGE FROM
                    CCCCXX '
8  MI       TEXT    'PASSING CONTROL
                    TO BBBBXX'
```

Here is a summary of the module disk names I used to set up this program package. You may change them if mine are confusing to you.

Main XBASIC module

DSK1.PT5/MAINXB

SCROLL and RESET modules

DSK1.PT5/SCROLL
DSK1.PT5/SCLOBJ
DSK1.PT5/RESET
DSK1.PT5/RESOBJ

Modules AAAAXX, BBBBXX & CCCCXX

(See Page 33)

LOADERS, MODULAR PROGRAMMING, LINKAGES & OVERLAYS

(Continued from Page 32)

DSK1.PT5/AAAAXX
DSK1.PT5/AAXOBJ
DSK1.PT5/BBBBXX
DSK1.PT5/BBXOBJ
DSK1.PT5/CCCCXX
DSK1.PT5/CCXOBJ

Here is the procedure for the whole package: Use the Editor/Assembler editor to create the source code for the five assembly modules and save them as follows:

DSK1.PT5/SCROLL
DSK1.PT5/RESET
DSK1.PT5/AAAAXX
DSK1.PT5/BBBBXX
DSK1.PT5/CCCCXX

Then load the assembler and create the five object modules and save them as follows:

DSK1.PT5/SCLOBJ
DSK1.PT5/RESOBJ
DSK1.PT5/AAXOBJ
DSK1.PT5/BBXOBJ
DSK1.PT5/CCXOBJ

Then load Extended BASIC and create the MAIN XBASIC module and save it as PT5/MAINXB.

EXECUTING THE SYSTEM

Load and run MAINXB. The message from modules AAAAXX, then from CCCCXX then from BBBBXX then from end of job should scroll up the screen.

For example:

Step 1. Load modules AAAA, BBBB and CCCC.

Step 2. Overlay CCCC only with module FFFF.

Step 3. Overlay BBBB and FFFF with a larger module EEEE.

Step 4. Finally, overlay AAAA and EEEE with module GGGG.

Do not forget that the REF/DEF table is a *stack* structure, working downwards in address locations. The module names must be cleared, as required, going upwards. The LSTLOW address pointer must be adjusted to reflect the un-stacking.

Consider the above imaginary procedure, in tabular form.

REF/DEF ADDRESS	ENTRIES STEP 1	ENTRIES STEP 2	ENTRIES STEP 3	ENTRIES STEP 4
>3FF8	AAAA	AAAA	AAAA	AAAA
>3FF0	BBBB	BBBB	EEEE	----
>3FE8	CCCC	FFFF	----	----
>3FE0	----	----	----	----

```
1            DEF     AAAAXX
2   AAAAXX   BLWP    @AA2
3            B       *R11
4   VMBW     EQU     >2Ø24
5   STAT     EQU     >837C
6   SCROLL   EQU     >AØØØ
7   MA       TEXT    'MESSAGE FROM AAAAXX '
8   M1       TEXT    'PASSING CONTROL TO CCCCXX'
9   OFSET    DATA    >6Ø6Ø
1Ø  WS       BSS     32

11  AA2      DATA    WS,AA4
12  AA4      LI      R3,MA
13           LI      R4,46
14  AA5      A       @OFSET,*R3+
15           DECT    R4
16           JNE     AA5

17           LI      R1,MA
18           LI      R2,2Ø
19           BL      @ALINE
2Ø           LI      R1,M1
21           LI      R2,26
22           BL      @ALINE

23           CLR     @STAT
24           RTWP

25  ALINE    MOV     R11,R1Ø
26           LI      RØ,>2EØ
27           BLWP    @VMBW
28           BL      @SCROLL
29           B       *R1Ø
3Ø           END
```

PROGRAM EXPLANATION (AAAAXX)

Line No.	Explanation
1	DEF module name
2-3	Start the run
4-6	Are EQUates needed
7-10	Are data and text items
11-16	Add offset value (=>60) to text messages to make them displayable in BASIC
17-22	Display the messages
23-24	Clear status and exit
25-29	Displays one line on screen and scrolls

Fig. 4

SEQUENCE OF OPERATIONS AND PROGRAM CODE

Here is the code and explanation for an Extended BASIC program that can be used to run the overlay system (all addresses and module lengths are assumed values) :

STEP 1 PROCEDURE

A. Module AAAA, relocatable, length >1A00 bytes, entry address (default) >24F4. Occupies all of low RAM space, >24F4 through >3EF4.

B. Module BBBB, absolute, AORG >A0000, length >2200 bytes. Occupies

(See Page 34)

LOADERS, MODULAR PROGRAMMING, LINKAGES & OVERLAYS

(Continued from Page 33)

high RAM space >A000 through >C200.

Module CCCC, absolute, AORG >C200, length >3600 bytes. Occupies high RAM space >C200 through >F800. No more high RAM space left.

D. Extended BASIC code.

```
1000 CALL LOAD("DSK1.AAAAOBJ
T")
1010 CALL LOAD("DSK1.BBBBOBJ
T")
1020 CALL LOAD("DSK1.CCCCOBJ
T")
```

E. REF/DEF table at this point.

```
>3FF8='AAAA ',>24F4
>3FF0='BBBB ',>A000
>3FE8='CCCC ',>C200
LSTLOW=>3FE8
```

F. XBASIC code continued.

```
1030 REM STARTING THE RUN IN
 MODULE CCCC
1040 CALL LINK("CCCC")
```

G. End of run for Step 1.

G1. Erase REF/DEF table entry for module CCCC.

```
1900 REM CCCC WAS AT >2FE8
1910 DEFF=3*4096+15*256+15*1
6+8
1920 CALL LOAD(DEFF,0,0,0,0,
0,0,0,0)
```

G2. Reset pointer in LSTLOW.

```
1930 REM LSTLOW IS AT >2004
1940 LSTLOW=2*4096+4
1950 REM NEED >3F F0
1960 G=3*16+15
1970 H=15*16
1980 CALL LOAD(LSTLOW,G,H)
```

STEP 2 PROCEDURE

A. Load module FFFF.

```
2100 REM LOAD MODULE FFFF
2110 CALL LOAD("DSK1.FFFFOBJ
T")
2120 REM MODULE FFFF LOAD IN
TO RAM AT >C200
```

B. REF/DEF table at this point.

```
>3FF8='AAAA ',>24F4
>3FF0='BBBB ',>A000
>3FE8='FFFF ',>C200
LSTLOW=>3FE8
```

C. Run FFFF.

```
2140 REM RUN FFFF
2150 CALL LINK("FFFF")
```

D. End of run of FFFF. Now we need to erase REF/DEFs for BBBB and FFFF.

```
2800 REM ERASE REF/DEF FOR B
BBB AND FFFF. START AT >3FF0
2810 DEFF=3*4096+15*256+15*1
6
2820 REM ERASE BBBB
2830 CALL LOAD(DEFF,0,0,0,0,
0,0,0,0)
2840 REM ERASE FFFF
2850 DEFF=DEFF-8
2860 CALL LOAD(DEFF,0,0,0,0,
0,0,0,0)
```

E. Reset LSTLOW.

```
2900 REM RESET LSTLOW, HERE
TO >3FF8
2910 REM LSTLOW POINTER AT >
2004. G STILL = >3F
2920 REM NOW NEED H = >3F
2930 H=15*16
2940 CALL LOAD(LSTLOW,G,H)
```

F. Status at this point. REF/DEF table.

```
>3FF8='AAAA ',>24F4
LSTLOW=>3FF8
```

STEP 3 PROCEDURES

A. Load module EEEE.

```
3100 REM LOAD MODULE EEEE
3110 CALL LOAD("DSK1.EEEEOBJ
T")
3120 REM MODULE EEEE LOADS I
NTO RAM AT >A000
```

B. REF/DEF table at this point.

```
>3FF8='AAAA ',>24F4
>3FF0='EEEE ',>A000
LSTLOW=>3FF0
```

C. Run module EEEE.

```
3140 REM RUN EEEE
3150 CALL LINK("EEEE")
```

D. End of run EEEE. Now erase REF/DEFs for AAAA and EEEE.

```
3800 REM ERASE REF/DEF FOR A
AAA AND EEEE
3810 DEFF=3*4096+15*256+15*1
6+8
3820 REM ERASE AAAA
3830 CALL LOAD(DEFF,0,0,0,0,
0,0,0,0)
3840 REM ERASE EEEE
3850 DEFF=DEFF-8
3860 CALL LOAD(DEFF,0,0,0,0,
0,0,0,0)
```

E. Reset LSTLOW, now back to >4000.

```
3900 REM RESET LSTLOW VALUE
TO >4000
3910 REM LSTLOW POINTER STIL
L >2004
3920 G=4*16
3930 CALL LOAD(LSTLOW,G,0)
3940 REM NOW RESET FSTLOW BA
CK TO >24F4
3950 G=2*16+4
3960 H=15*16+4
3970 REM LSTLOW POINTER IS >
2002, = 8194
3980 CALL LOAD(8194,G,H)
```

STEP 4 PROCEDURE

A. Load module GGGG. GGGG is relocatable, entry at >24F4, length >1A00 bytes, occupies low RAM >24F4 through >3EF4.

```
4000 REM LOAD GGGG
4010 CALL LOAD("DSK1.GGGGOBJ
T")
```

B. REF/DEF table at this point.

```
>3FF8='GGGG ',>24F4
LSTLOW=>3FF8
```

C. Now run GGGG.

```
4150 CALL LINK("GGGG")
```

D. Run of GGGG terminates the operation.

Fourth annual Seattle TI99/4A meet set

The fourth annual TI99/4A Seattle Convention is scheduled for Sept. 23-24 in Kenmore, Washington.

Location is the Kenmore Flea Market in Kenmore Square, open from 10 a.m. to 4:40 p.m. each day.

To reserve vendor space, limited to 12 tables, call Barb Wiederhold at (206) 361-0799 (voice) or (206) 361-0895. Cost is $15 per table.

Admission to the flea market is 50 cents at the door and admission to the convention area is $2.

According to Cynthia Becker, a convention organizer, attendees may bring disks or buy them at the user group booths. Two tables will be reserved for user groups, who need to register for space also, she says.

A Friday night reception is scheduled in connection with the convention. Tickets are $2.50 per person. A Saturday night bowling party is scheduled at $10 per person. Both these events are for adults only.

"Slumber party" accommodations are available for convention attendees, with males in an apartment recreation room and women in the apartment across the hall. Participants need to register for these accommodations by contacting Wiederhold at the above telephone numbers, according to Becker.

A CHARACTER GENERATOR FOR GENEVE & TI99/4A

Buffer of 99/4A can't handle this program in one bite

This is the second of three installments of CHARA1FIX, which began last month. The program, by Wayne Stith, is used to redefine characters in the CHARA1 file used with TI-Writer, MY-Word and other programs. After creating characters, the program saves the result as a CHARA1 file.

The assembly language program requires a CHARA1 file, memory expansion and disk system. The program code is entered using MY-Word, TI-Writer or the Editor-Assembler editor. The program runs on the 99/4A as well as the Geneve. However, it is not compatible with CorComp disk controllers at this time. MICROpendium will publish a patch for CorComp users when it becomes available. An explanation of how to use the program was published last month.

Because of the length of the program as it appears here, it cannot be loaded completely into the Editor/Assembler buffer using the 99/4A. However, the author has outlined two methods that may be used to compensate for this.

METHOD ONE

With this method, the user will save the program as two files. Type in as much of the program as possible using the Editor/Assembler editor, TI-Writer or Funnelweb. When the buffer is full, delete the last few lines to gain buffer space and add the following line:

COPY "DSK1.FILE2"

or whatever the second file will be called. Then save the current buffer contents as DSK1.FILE1, or whatever name you decide to use. Then purge the buffer and pick up with the source code where you left off and enter it and save it using FILE2, or whatever name you selected.

If the buffer fills up again, repeat the procedure using DSK1.FILE3, etc. The source code files will be chained in this way.

Assemble the file DSK1.FILE1.

METHOD 2

This is the way the author does it when he has a gigantic source file to type:

Create a "copier" or master file like this:

```
COPY "DSK1.FILE1"
COPY "DSK1.FILE2"
COPY "DSK1.FILE3"
```

Use as many filenames as you need. Then type in one-third of the source code and save it as DSK1.FILE1, the second third as DSK1.FILE2, etc. Assemble the file DSK1.COPIER (or whatever you called the master file).

An extension to this idea is the following:

```
      DEF   SFIRST,SLAST,SLOAD,START (or whatever)
SFIRST
SLOAD
START
            COPY "DSK1.FILE1"
            COPY "DSK1.FILE2"
            COPY "DSK1.FILE3"
SLAST       END
```

The latter extension requires a bit of careful looking at the source code to make sure that the DEF and various label statements are not duplicated anywhere.

Regardless of the method, the END statement should appear *after* all the source code will have been "seen" by the assembler. With the COPY directive (see E/A manual) the TI assembler is a powerful tool, as it can assemble dozens of files and generate many more K of code than the TI can actually handle.

CHARA1FIX

```
0554
0555  * Move up.
0556
0557  UP    MOV  @LINE#,R5    On line 1 ?
0558        JEQ  UP1          Yes
0559
0560        AI   R1,-32       No, subtract 32 from cursor locaton
0561        DEC  @LINE#       Adjust grid line pointer
0562        JMP  PRESC1       Start over
0563
0564  UP1   MOV  @SEVEN,@LINE#     Set pointer to line 7
0565        AI   R1,7*32      Point to a position on bottom line
0566        JMP  PRESC1       Start over
0567
0568  * Move down.
0569
0570  DOWN  MOV  @LINE#,R5
0571        CI   R5,7         On bottom line ?
0572        JEQ  DOWN1        Yes
0573
0574        INC  @LINE#       Add 1 to grid line pointer
0575        AI   R1,32        Add 32 to cursor location
0576        JMP  PRESC1       Start over
0577
0578  DOWN1 CLR  @LINE#       Set pointer to line 1
0579        AI   R1,-224      Point cursor to a position on top line
0580        JMP  PRESC1       Start over
0581
0582  * FCTN-7 was pressed for the help screens...
0583
0584  FCTN7 CLR  @>83C4       Kill ISR hook
0585        BLWP @HELP        Call HELP routine
0586        LI   R3,MYINT1    Restore interrupt
0587        MOV  R3,@>83C4
0588        JMP  PRESC1       Pick up where we left off
0589
0590  * (ENTER) pressed.  Change white space ('pixel') to black or vice versa.
```

(See Page 36)

CHARA1FIX—

(Continued from Page 35)

```
0591
0592 ENTER  BLWP @VSBR        Read the current screen location
0593
0594 * The possible codes returned to R1 are >00 (white) and >08 (black)
0595
0596        MOVB R1,R1        White space ?
0597        JEQ  ENT1         Yes
0598        CLR  R1           No, load >00
0599        JMP  ENT2
0600
0601 ENT1   LI   R1,8*256     Load >08 (8*256 is the same as >0800)
0602 ENT2   BLWP @VSBW        Write new value
0603
0604 * Update screen displays elsewhere
0605
0606        BLWP @BOXRD       Read the grid's values
0607        BLWP @LITTLE      Show little character
0608        BLWP @HEXDIS      Show new hex string
0609        JMP  RIGHT        Move the cursor one space to the right
0610
0611 * FCTN-6 pressed, show the next sequential character
0612
0613 FCTN6  INC  @CURCHR          Increment current character number
0614        C    @CURCHR,@HIGHCH  Too high ?
0615        JLE  FCTN6A           No
0616
0617        CLR  @CURCHR          Yes, start at ASCII 0
0618 FCTN6A B    @MAIN1           Start over with fresh screen
0619
0620 * FCTN-4 pressed, show the next lower character
0621
0622 FCTN4  MOV  @CURCHR,R5
0623        DEC  R5           Decrement current character number
0624        CI   R5,256       Gone too far ? (Subtracting 1 from 0 would be >FF
FF)
0625        JLE  FCTN4A       No
0626
0627        MOV  @HIGHCH,R5   Yes, start at highest allowable character
0628
0629 FCTN4A MOV  R5,@CURCHR   Store new value
0630        JMP  FCTN6A       Update entire screen with new character
0631
0632 * FCTN-9 pressed, switch to hex mode after setting flag
0633
0634 FCTN9  SETO @MODE        Set flag
0635
0636 * The following routines handle keypresses while in the hex string
0637
0638 MODE2  LI   R0,520       Point to left end of hex line
0639 PRESC2 BLWP @VSBR        Read in current value
0640
0641        MOV  @FIVE,@INTREG+2  Set up interrupt timer
0642        LI   R3,MYINT1
0643        MOV  R3,@>83C4    Load ISR hook
0644
0645 SCAN2  BL   @KK          Scan keyboard
0646
0647        BLWP @VSBW        Display current value
0648        MOVB @>8375,R3    Fetch new value
0649
0650 * Check for acceptable codes
0651
0652        LI   R6,CODES     Point to list of codes
0653        LI   R7,JMPTB2    Point to branch-address table
0654
0655 CHECKA MOVB *R6+,R2      Fetch code to check
0656        JEQ  CHAR         If zero, entire list has been checked
0657
0658        CB   R3,R2        Code match ?
0659        JEQ  CHECKB       Yes
0660        INCT R7           No, add 2 to pointer because this is DATA
0661        JMP  CHECKA       Check some more
0662
0663 CHECKB MOV  *R7,R7       Fetch table address
0664        B    *R7          Branch to that address
0665
0666 * The keypress must be some other character; make sure it is within
0667 * the proper range.
0668
0669 CHAR   LI   R4,HEX        Point to displayable hex code table
0670 CHAR1  CB   R3,*R4+      Match ?
0671        JEQ  CHAR2        Yes
0672        CI   R4,HEX+16    Gone far enough ?
0673        JNE  CHAR1        No
0674        JMP  SCAN2        Yes, scan keyboard again
0675
0676 CHAR2  MOVB R3,R1
0677        BLWP @VSBW        Display new value
0678
0679        BLWP @HEXWR       Update hex string
0680        BLWP @BOXWRT      Update grid
0681        BLWP @LITTLE      Show little character
0682        JMP  RX           Move cursor to the right
0683
0684 * Move left.
0685
0686 LL     CI   R0,520       Already at the far left ?
0687        JEQ  PRESC2       Yes, scan again
0688        DEC  R0           No, subtract 1 from cursor location
0689        JMP  PRESC2       Scan again
0690
0691 * Move right.
0692
0693 RX     CI   R0,520+15    Already at the far right ?
0694        JEQ  PRESC2       Yes, scan again
0695        INC  R0           No, add 1 to cursor location
0696        JMP  PRESC2       Scan again
0697
0698 * FCTN-7 pressed for help screens
0699
0700 F7     CLR  @>83C4       Kill ISR hook
0701        BLWP @HELP        Call HELP routine
0702        LI   R3,MYINT1
0703        MOV  R3,@>83C4    Restore ISR hook
0704        JMP  PRESC2       Scan again
0705
0706 * FCTN-6 pressed, show next higher character
0707
0708 F6     INC  @CURCHR          Increment current character number
0709        C    @CURCHR,@HIGHCH  Higher than allowed ?
0710        JLE  F6A              No
0711
0712        CLR  @CURCHR          Yes, start with ASCII 0
0713 F6A    B    @MAIN1           Start over with fresh screen
0714
0715 * FCTN-4 pressed, show next lower character
0716
0717 F4     MOV  @CURCHR,R3
0718        DEC  R3           Subtract 1 from current character number
0719        CI   R3,256       Gone too far ?
```

(See Page 37)

CHARA1FIX—

(Continued from Page 36)

```
0720        JLE  F4A          No
0721
0722        MOV  @HIGHCH,R3   Yes, start over with highest allowable character
0723
0724  F4A   MOV  R3,@CURCHR   Store new value
0725        JMP  F6A
0726
0727  * FCTN-9 pressed, switch to grid mode
0728
0729  F9    CLR  @MODE        Clear flag
0730        B    @MAINX       Start over
0731
0732  * CNTL-R pressed from either mode; restore original character pattern.
0733  * Multiply the character number by 8 to obtain an offset, then add
0734  * >2000 so that R0 points to the location in VDP where the original
0735  * pattern is stored from the loaded CHARA1 file. Add >E000 to R1
0736  * so that it points to the CPU RAM storage location for this same
0737  * character's pattern.
0738
0739  CNTLR MOV  @CURCHR,R0
0740        MPY  @EIGHT,R0
0741        MOV  R1,R0
0742        AI   R0,>2000
0743        AI   R1,>E000
0744        LI   R2,8
0745        BLWP @VMBR        Read 8-byte pattern
0746        B    @TRY         Start over with a fresh screen
0747
0748  * CNTL-W pressed from either mode; write CHARA1 back to disk.
0749
0750  CNTLW LI   R0,576
0751        LI   R1,MSG7
0752        LI   R2,32
0753        BLWP @VMBW        'Ready to write...'
0754
0755        AI   R0,32
0756        A    R2,R1
0757        LI   R2,27
0758        BLWP @VMBW        ' Press <ENTER>...'
0759
0760        AI   R0,32
0761        A    R2,R1
0762        INC  R2
0763        BLWP @VMBW        'or any other key...'
0764
0765        LI   R0,599
0766        MOVB @DRIVE,R1
0767        BLWP @VSBW        Display drive number in message
0768
0769        CLR  @>83C4       Kill ISR hook to avoid anything blinking
0770
0771        BL   @KK          Wait for a keypress
0772
0773        CB   @>8375,@CODES+1  <ENTER> ?
0774        JEQ  CNTLW1           Yes, continue
0775        B    @TRY             No, start over with a fresh screen
0776
0777  CNTLW1 LI  R0,>3500     Send output PAB to VDP
```

(See Page 38)

CHARA1FIX—

(Continued from Page 37)

```
0778          LI   R1,CPAB2
0779          LI   R2,21
0780          BLWP @VMBW
0781
0782          LI   R0,>3500+13
0783          MOVB @DRIVE,R1
0784          BLWP @VSBW          Insert drive number
0785
0786          LI   R0,>2000
0787          LI   R1,>E000
0788          LI   R2,>1800
0789          BLWP @VMBW          Write altered character patterns back to VDP
0790
0791  * The PAB needs to know the number of bytes from VDP it should write to
0792  * the file.  Since CHARA1 files vary in length, we must calculate this on
0793  * the basis of the number of sectors the file originally contained.
0794  * This information will be stored in the header and the PAB.
0795
0796          MOV  @TOTSEC,R3
0797          MPY  @X256,R3
0798          LI   R0,>1FFC
0799          LI   R1,REGS+8      This is R4
0800          LI   R2,2
0801          BLWP @VMBW
0802
0803          LI   R0,>3500+6
0804          BLWP @VMBW          Place the bytes in the PAB
0805
0806          LI   R3,>3509
0807          MOV  R3,@>8356      Point to the length byte in the PAB
0808          BLWP @DSRLNK        Write the data out to disk as CHARA1
0809          DATA 8
0810
0811          JEQ  BOTCH
0812          B    @START         If no error, start program over
0813
0814  * Disk error
0815
0816  BOTCH   LI   R0,576         Clear part of screen
0817          LI   R1,>2000
0818  BOTCH1  BLWP @VSBW
0819          INC  R0
0820          CI   R0,704
0821          JNE  BOTCH1
0822
0823          LI   R0,587
0824          LI   R1,MSG6        'I/O error...'
0825          LI   R2,10
0826          BLWP @VMBW
0827
0828          LI   R0,617
0829          LI   R1,MSG4        'Press any key...'
0830          LI   R2,15
0831          BLWP @VMBW
0832
0833          BL   @KK            Wait for key
0834  BOTCHX  B    @TRY           Start over without losing data
0835
0836  * HELP screen routine
0837
0838  HELP    DATA >8300,HELP1    Vector;  use scratchpad for workspace
0839  HELP1   LI   R0,>0205
0840          BLWP @VWTR          Change screen image table to >1400
0841          BL   @CSS           Clear this new screen
0842
0843          LI   R4,4           Loop counter
0844          LI   R1,HMSG1       Point to first page of messages
0845
0846  HELP1A  BL   @SETT          Set VDP address to top left corner of screen
0847          LI   R0,20*32       Total of 640 bytes to send to screen
0848  HELP2   MOVB *R1+,@>8C00    Send a byte
0849          DEC  R0             Done ?
0850          JNE  HELP2
0851
0852          BL   @KK            Wait for a key
0853          DEC  R4             Decrement loop counter
0854          JNE  HELP1A         Done ?
0855          LI   R0,>0200       Yes, restore screen image table to >0000
0856          BLWP @VWTR
0857          RTWP                Go home
0858
0859  * SETT sets the VDP write address to >1400. The first two bits
0860  * must be set to 01, which yields an address of >5400.
0861
0862  SETT    LI   R0,>0054
0863          MOVB R0,@>8C02
0864          SWPB R0
0865          MOVB R0,@>8C02
0866          RT
0867
0868  * Clear screen at VDP >1400.
0869
0870  CSS     LI   R0,>0054       VDP address with first bits adjusted,
bytes reversed
0871          MOVB R0,@>8C02      Set address
0872          SWPB R0
0873          MOVB R0,@>8C02
0874          LI   R0,768         Do this 768 times
0875  CSS1    MOVB @SET,@>8C00    Send a space to the screen
0876          DEC  R0
0877          JNE  CSS1           Done ?
0878          RT                  Go home
0879
0880  * Clear screen at >0000
0881
0882  CS      LI   R0,>0040
0883          MOVB R0,*R13        Since this routine is called only from
0884          SWPB R0             the normal workspace, we can use the
0885          MOVB R0,*R13        addresses in R13 and R14
0886          LI   R0,768
0887  CS1     MOVB @SET,*R14
0888          DEC  R0
0889          JNE  CS1
0890          RT
0891
0892  * Scan keyboard
0893
0894  KK      BLWP @KSCAN
0895          MOV  @>837C,R3      Fetch GPL status
0896          LIMI 2              Let interrupts in briefly
0897          LIMI 0
0898          COC  @SET,R3        Key pressed ?
0899          JNE  KK             No
0900          CB   @>8375,@SPLIT  CNTL-= ?
0901          JEQ  QUIT           Yes
0902          RT                  No
0903
0904  * QUIT Key (CNTL-=) for Geneve users who don't like CNTL-SHIFT-SHIFT
```

The final installment of this program will appear next month.— Ed

Page Pro 99

Easy way to integrate text, graphics

By HARRY BRASHEAR

In the world of TI graphics, we all have a different idea of what the perfect page maker WILL be.

Quite a few darn good entries have come into this race in the past few years. FontWriter, CSGD, Jiffy Flyer, Form Shop, Picasso and Printers Apprentice, just to name a few, have wowed us every time. Each piece, in it's own right, has been fabulous, but...

There is almost always a "but," or an "if" in the bottom line. No single item is ever going to please everyone, or give the user all of what he is going to need forever.

Fortunately for us, the software keeps coming, getting better, more inventive, and easier to use all of the time. So, ladies and gentlemen, may I present this year's winner in the "Easiest to use—one shot page maker" category, Page Pro 99.

First, take a look at the sample page I made up for this review. Yeah, I know, it's a nightmare, a hodgepodge of pictures, text, and bad humor, but it gives you a good idea of what can be done with this program. There are few restrictions to format, (obviously) and you can make it look as pro as you want.

First, let's take a look at the major features of Page Pro.

- It allows you to design one complete 8½ x 11 page in one shot and save or print it out.
- It has full screen editing. This includes most of the editing features of TI-Writer, in the familiar key presses.
- It allows border making in all of the IBM ANSI graphic characters.
- You may add up to 28 pictures of any size anywhere you like on the page. The pictures are converted from any TI-Artist instances you may have, or wish to create.
- Two font sizes, one for text, one for titles, may be selected from various styles. Both sizes have full editing features.

OK that should have grabbed you, now let's look at the amazing details.

Page Pro will allow you to make up the entire 8½ x 11 page in just one setting by giving you the ability to page up, down and sideways, showing you 12 lines by 30 columns at a time. Your final format size is 66 lines by 60 columns. The full page format is tight. If you want to use the full 66 lines, you *have* to start your printer just a hair off the perf. The side borders are only about 3/8-inch, so centering or proper page alignment is a must.

Using Page Pro as a text editor is, for all practical purposes, the same as using one of the Writers. Cursor movement is handled with the FCTN arrow keys to move around the page. You may delete a character, or line, insert a character or line, page up, down and sideways, all with exactly the same keys as the Writers use. All the editing functions work when you are using regular text size, but a few line editing limitations exist in double size, along with a minor slowdown in cursor speed. It's not bad, but you will miss a letter if you don't cut your typing speed by about half. No big deal!

The cursor will wrap to the next line, as any word processor, but you do not have word wrap. I didn't mind this so much because it gave me a chance to hyphenate. Also, if you want to hand justify for the right column, this is easy to do also. Speaking of justification, if you want to, you can prepare your text file outside the Page Pro environment with any of the Writers, and then just import the text to the Page Pro screen. Likewise, you can save the text from Page Pro to a DV/80 file for Writer use.

One of the more interesting edit functions is the ability to type in any direction. By using CTRL arrow keys, you change the path of the cursor to the arrowed direction. This is cute for various special word effects, but it also has a practical side. When you are using the border function, you can use this method to run lines down, across and up the other side, efficiently.

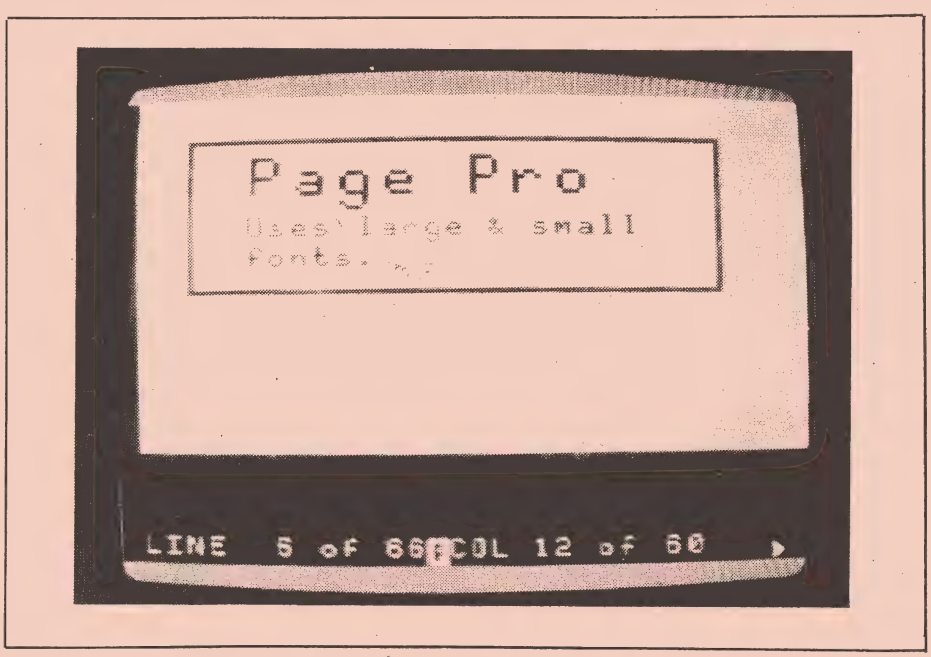
(See Page 40)

Review

Report Card

Performance	**A+**
Ease of Use	**A+**
Documentation	**B+**
Value	**A**
Final Grade	**A**

Cost: $24.95 plus $1.50 postage
Manufacturer: Asgard Software, P.O. Box 10306, Rockville, MD 20850
Requirements: memory expansion, disk system, printer, 99/4A or Geneve

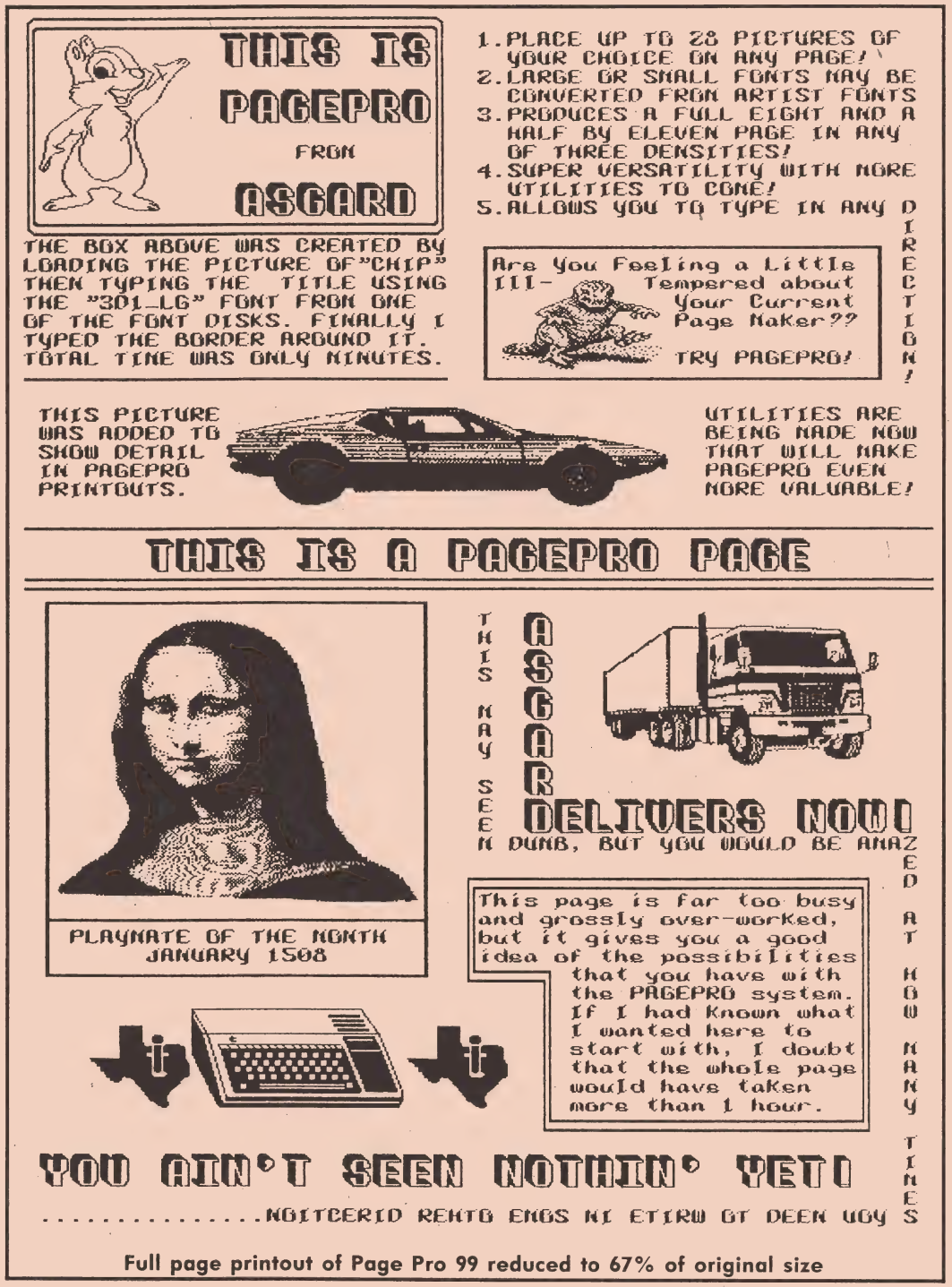


PAGE PRO 99—

(Continued from Page 39)

By pressing CTRL "8" you are put into "LINE mode." I prefer to call this function "BORDER mode" because that's its main use. Thirty characters represent IBM graphic characters. (But you don't need that mode on your printer to use them in the output.) They can be used as underscores, boxing in the pictures, or even putting a border around the entire page. All the editing functions apply to this phase, just as they do in text mode. When in use, there is a cheat sheet that appears in the bottom half of the screen to assist in the proper key presses.

The crowning glory of Page Pro is its ability to use pictures anywhere on the page, up to 28 of them. These pictures are conversions from the TI-Artist instance format to one better used by Page Pro. "Oh no," you say, "another graphic format?" Yes, but there is a darn good reason for it. First of all they wanted a compressed format that would load quickly, and secondly, one that would take less file space. For comparison, a full page in Picasso, (two Picasso files) requires 170 sectors, a Page Pro full page file needs 21 sectors. *Quite* a difference! To help you, an Extended BASIC program is included with Page Pro to convert Instances to the proper format. It's slow working, but you only have to do it once anyway.



Full page printout of Page Pro 99 reduced to 67% of original size

For the lazy among you, or for those that don't use Artist, Asgard is selling seven disks of pictures already converted to get you going. Because of the compressed format, quite a few pictures ares on each disk. Actually, they work out to be a better value, per picture, than their old Artist companions.

To load a picture to the page, you just place the cursor at the top left corner of where you want the picture to be. Press CTRL "L" and type in the device and picture name. The picture is loaded in an "as is" mode, in other words, you see the picture just as it is, and will be, in the final printout. However, once you are satisfied with its placement etc., you should press CTRL O and turn it off. The "turning off" process simply changes the turned on pictures to inverse P's. This is done so that the speed of editing can be maintained. If you have to leave the picture on and move past the visible screen area, the program has to reload part of the picture. This then becomes a slow process in moving around. The whole concept is really pretty well

(See Page 41)

PAGE PRO 99—

(Continued from Page 40)

thought out.

The pictures are *not* kept in the page file. The program simply notes the name of the picture and what disk drive it came from in the file, along with the text. When you want to reload the page it goes back to the respective drive or drives for the pictures. I couldn't resist using my Horizons for this process just once. Boy, did that stuff load quick! The only problem is that if you forget to put the given picture in the right drive, you get an I/O error. My best advice is to get organized right away with this program. Put everything you need to do a page in one drive, and don't be chasing from one to the other. You'll learn that lesson the hard way, I'm sure.

The program will access up to 28 picture files, so you should never be at a loss for placing graphics on a page. You can also erase any given picture with CTRL K if you misplace it.

USING THE FONTS

The defaulted large and small fonts are nice, but a little blocky. You have the ability to convert other Artist fonts over to PP format, again, with a conversion program included. A small PP font is eight pixels wide by 12 pixels high. A large one is 24 high by 16 wide, so you should have an idea of what you're dealing with before you start converting. Conversion is easy, a little slow again, but well worth the time spent. I have done this to a few of my Artist fonts and found it quite painless. I repeat, you only have to do it once.

Since many of you may be a little short on Artist stuff, don't worry. Paul Scheidemantle, one of our best graphic designers, has been commissioned by Asgard to produce about 50 custom fonts for this program. By the time you read this, they should be available.

Once your page is all set, you press CTRL P to send it to your printer. You have the option of single, double and quad densities, so you can do a quick dump, or a nice copy for duplication purposes. The printout is fast enough even though the system is accessing all of those picture files. I found the line feeds perfectly acceptable too. This is important to keep from getting those dumb little white skip lines in solid black areas. If you just want a quick dump to see how things line up, then you can keep the pictures turned off during printing. You can still see some dots in the text printing, but it's not bad. A camera-ready print function, similar to Bob Coffey's famous Instance printer, is promised for one of the forthcoming utility disks.

A columnizer utility is included with the program that will set up two-column pages of text for you. The nice part of this one is that it sets up your TI-Writer files in consecutive page files. In other words, if you have a lot of text, it sets it up as PAGE01, PAGE02, PAGE03, etc. Each file will be one Page Pro page, and can be loaded into the program with CTRL F''. If you expect to use pictures, though, you will have to leave room for them in the text. It's a nice little program, and comes in handy for newsletters. It doesn't care what your format is, from 25 to 80 columns, it sets it up in two 28-column rows as neat as a pin. It will justify and page number, (top or bottom of page) optionally.

The documentation included with the program is good. It consists of two manuals, one to get you started, and another for the utilities and advanced techniques, 35 pages in all. They are well laid out, and both the TI beginner and the TI hacker will benefit from reading them. In all fairness though, a file called QUICKREF that can be loaded into the program and printed out, tells as much as you need to know to get started. The program itself is on one disk, but a second disk is included that's loaded with test files and examples to work with.

Two versions of the program are included, one for the 4A and another one specifically for the 9640. I am told that there are no real differences, though. It mainly had to do with the VDP I guess, so they had to do it twice. Nuff said that nobody gets left out of this one. All in all, it's a nice program with a lot of potential. The price is right at $24.95 and I have already used it on a couple of our newsletters with good results. Ed Johnson, the author of Page Pro, is an excellent assembly programmer and he has a lot of good ideas for future PP utilities. It works well, I couldn't find any nasty bugs, and it does exactly what it says it will. Who could ask for more? I haven't seen potential like this for a long time and it's geared to the end user... about as friendly and easy as they come. All you have to do is read the docs and you can produce. If you are looking for a publishing system, dive in now. There's a lot more to come, I think, and it's not a program to be afraid of.

Starlink system provides BBS calling

A new system on the Tymnet service, StarLink, is a service which provides linkups via modem to long distance bulletin boards, bypassing long distance charges, similar to Telenet's PC Pursuit.

Information and online signups are available via modem at 1-800-343-3704.

The signup information also contains information on the Galaxy Information Network BBS.

More than 130 United States cities can be reached through StarLink's outdial service and StarLink has more than 1,000 local access numbers available in the United States and Canada.

Fees include $25 registration, a monthly $10 service fee, and $1.50 per hour in the ''lower 48'' continental United States during non-prime time hours (7 a.m.-6 a.m. Monday through Friday, all day Saturday, Sunday and major holidays).

Membership entitles the user to two free hours on the Galaxy Information Network (StarLink connect charges still accrue).

Persons dialing the information number are required to type in name, street address and home telephone number before the service provides further information.

Newsbytes

Disk Only Software gets new 800 number

Disk Only Software has installed a new toll-free 800 number to handle credit card orders 24 hours a day, seven days a week, according to Jeff Guide of the company.

Disk Only Software accepts Visa, MasterCard and American Express, Guide says.

The number, 1-800-736-4951, currently operates in the continental United States. Guide says the company plans to add toll free 800 number phone service from Canada in the near future. Orders from Canada and overseas may be placed by calling (703) 550-9877, he says.

Guide says the company plans to phase out its previous toll free 800 number because of an increase in phone charges.

He says Disk Only Software has added a direct order line for ordering or checking availability of Myarc computer products at (301) 340-7179, 9 a.m.-5 p.m. EDT.

The company also deals in Genial Computerware, Asgard Software and Horizon RAM Disks.

Address for Disk Only Software is P.O. Box 244, Lorton, VA 22079.

Disk index offer no longer available

Norberto Bettinelli of Buenos Aires, Argentina, is no longer able to provide the MICROpendium index on disk he previously offered.

His son, Leando Bettinelli, advises that his father has been bedridden for the last two months and is expected to remain so for several more.

WINDYXB offered

Richard Lynn Gilbertson is offering an extension to Extended BASIC for the TI accessed through CALL LINK statements in XBASIC, called WINDYXB.

The program may be downloaded from the PUNN (Portland Users of 99s) BBS at (503) 233-6804. Gilbertson is offering the program as fairware, and asks that it be distributed by users' groups only.

Extensions include:

1. CALL LINK("HPUT",ROW,COL,STRING)

This link duplicates the DISPLAY AT in Extended BASIC, but uses columns 1 to 32 and rows 1 to 24 for horizontal displays.

2. CALL LINK("VPUT",ROW,COL,STRING)

This link duplicates the DISPLAY AT in Extended BASIC, but uses columns 1 to 32 and rows 1 to 24 for vertical displays.

3. CALL LINK("HGET",ROW,COL,LENGTH,STRING)

This link retrieves a string from the screen horizontally as specified in LENGTH.

4. CALL LINK("VGET",ROW,COL,LENGTH,STRING)

5. CALL LINK("WINDOW",WINDOW#(1-6),NUMBER)

This link either stores or loads the entire screen to or from memory. (Sprites and colors are not saved.)

6. CALL LINK("WINDY",ROW#(1-21),NUMBER)

This link either stores or loads the entire screen to or from memory. (Sprites and colors are not saved.) (6 window * 24=121 rows.)

7. CALL LINK("SDUMP","DSK#.file name")

This link stores to disk everything on screen: sprites definitions, colors speed, location of sprites and patterns, colors, positions of screen (except screen background color).

8. CALL LINK("LDUMP","DSK#.file name")

This link loads to screen and memory what was dumped with SDUMP and 1.8 seconds later the screen is initialized fully. The sprites are moving and all colors (except screen background color).

9. CALL LINK("DUMP",TAB#)

This link dumps to printer as a standard screen dump utility, with the tab number indicating tab position on paper (PIO Epson compatible only).

Gilbertson says he has created a 100 percent compatible program with RAM disks using a standard DSR link tested on Myarc and CorComp RAM disks. He says XBASIC programs can be rewritten using WINDYXB to take the place of numerous items. He says he has "knocked off as much as 5K from an original 12K game program, plus received the added bonus of the program running considerably faster."

For further information, contact Gilbertson at 2205 S.E. Salmon, Portland OR 97214.

Software released

DDI Software has released Appointment Scheduler for the Geneve 9640 and a utility package called Artist Utility.

Appointment Scheduler requires MDOS 95H and Advanced BASIC 6/29 or later versions.

The program is designed to create an appointment schedule on a day-to-day basis for every month of the year. According to the manufacturer, it uses many of the new commands available, such as TCOLOR, redefinitions of command line, CALL MARGINS and CALL CHAR above ASCII 143.

Artist Utility includes Slidemaker, a program to create slides and instances for TI-Artist; Prografix, a program to create graphics for use in a program in HEX or ASCII; and Textgrafix, a program to create small graphics for My-Word and TI-Writer. It uses no TLs and runs out of Extended BASIC.

Appointment Scheduler sells for $15 and Artist Utility for $12. For further information or to order, contact DDI Software, 2004B Leann, Austin, TX 78758.

Videotapes available to users groups

Videotapes of all seminars at the May 20 Lima Ohio Multi User Group conference are available to any user group for the cost of media and postage, according to Charles Good, librarian for the group.

The tapes are not available to individuals, except paid members of the Lima User Group.

The tapes contain almost 11 hours of material, including presentations by Barry Traver, Chris Bobbitt, Bud Mills, Jim Horn and Paul Scheidemantle.

Also shown are demonstrations of SUPERBASIC v2, MX-DOS v3.0 and some features of an as-yet-unreleased major revision of Funnelweb designed specifically for 80-column systems.

User groups can obtain copies of these

(See Page 43)

Newsbytes

(Continued from Page 42)

tapes by sending two VHS videotapes and a paid return mailer, or $10, to the Lima Ohio User Group, P.O. Box 647, Venedocia OH 45894. Those groups in the U.S. sending money will receive their videotapes via fourth class mail, Good says.

More on TI-Tref

The Fourth Annual TI-Tref is scheduled for Oct. 7 in the Het Kolpinghuis, located within five minutes walking distance from the railway station in Nijmegen, The Netherlands. The three previous events were held in Germany.

According to Drs. E.C. van Wette, secretary of the TI-Gebruikersgroep, the Dutch users group, 40 European users groups have been invited to the meeting.

Presentations planned include one by a Myarc representative, van Wette says. Items will be available for sale, an auction will be held and door prizes will be awarded.

For further information, contact Vereniging TI-Gebruikersgroep, Secretariaat: Drs. E.C. van Wette, Kremersmaten 106, 7511 LC Enschede, The Netherlands.

Texaments releases new products

Texaments has released TI Sort and four new companion products for TI Artist.

TI Sort is described as the only universal sort utility available for the TI99/4A.

Steve Lamberti, president of Texaments, says TI Sort is a standalone, fully menu-driven program capable of sorting an unlimited amount of records stored in almost any type of file, including TI Base files, delimited files, fixed length files and BASIC or Extended BASIC files. Files to be sorted may be located on any type of device, including floppy disk, RAMdisk and hard disk, he says.

TI Sort is described as being based on the "Quick Sort" method originally developed by C.A.R. Hoare, and as occupying a minimum amount of system memory. It will sort on up to eight fields and sorts large blocks of data in a dual pass process, according to the manufacturer.

In time trials, Lamberti says, a standard TI Base files with 843 records sorted on two fields was completed in 8 minutes, 59 seconds using a standard floppy drive. The same sort was completed in three hours, 47 minutes and 26 seconds using the SORT directive of TI Base. A standard delimited 999-sector file with 2990 records sorted on one file was completed in 13 minutes, 49 seconds using a hard disk drive.

Operating requirements for TI Sort include a disk system, 32K memory expansion and either Extended BASIC, Editor/Assembler or Mini-Memory. The utility is compatible with all storage devices and the Geneve 9640 in GPL mode. Cost is $14.95.

The Artist's Companion disks 10-13 have added predefined scenes to the series, designed for users to create pictures and backdropw or to accent existing drawings. Scenes consist of small "building blocks" which may be arranged in any manner on a horizontal plane. Scenes include fences, mountains, skylines and beaches, Lamberti says.

Also new to the series are predefined borders, including beads, ants, hearts and ribbons.

The two-disk sets also each contain an assortment of fonts and small and large instances.

Artist's Companion #10 contains six fonts, 13 large instances, seven small instances, nine borders and five scenes; #11 contains six fonts, 13 large instances, seven small instances, 10 borders and five scenes; #12 contains six fonts, nine large instances, 10 small instances, 14 borders and four scenes; and #13 contains three fonts, eight large instances, 11 small instances, 25 borders and three scenes.

The new Artist's Companion sets are available for $9.95 each, any two for $17.90 any three for $26 or all four for $34.

A $2.50 shipping charge applies to all orders.

For information or to order, contact Texaments, 53 Center St., Patchogue, NY 11772 or (516) 475-3480.

User Supported Software

ZODIAC WHEEL OF FORTUNE, NIGHT SNIPER

These two programs require XBASIC, disk system and memory expansion. Zodiac is a fortune-telling program that displays zodiac signs based on dates, forecasts lucky periods and makes note of important events. The cost is $5.

The object of Night Sniper is to fight invading aliens under cover of darkness. Your weapon is a night-penetrating gun scope. The author is John Bulakowski. The cost is $4.

Send disk and postage-paid return mailer to Janet Ryan, 10 Jolly Rd., Ellington, CT 06029. Proceeds are to be donated to the Nutmeg TI-99ers.

JAPANESE

Japanese is a diskful of programs designed to help the student of the language, but can also be useful to show off the TI at its best. An XBASIC loader will walk you through the whole disk. Six pages of documentation are included on a D/V80 file. Programs cover all three forms of written Japanese, as well as five printer programs that make study sheets in Romanized Japanese. One of the programs requires memory expansion, the remainder run on a console and disk drive. Cash donations are accepted after delivery. Send disk, self-address, postage-paid return mailer to Don Shorock, Box 501, Great Bend, KS 67530.

***User Supported Software* is non-commercial software written and distributed by readers. Anyone wishing to submit an announcement is encouraged to send a copy of the program as well as a description to MICROpendium, P.O. Box 1343, Round Rock, TX 78680. MICROpendium cannot take responsibility for any announcement that appears in this column. Readers who encounter as a result of placing orders are asked to contact us by mail. An inclusive listing of *User Supported Software*, consisting of 10 pages, is available from MICROpendium for $2.**

User Notes

AUTOEXEC and menu for hard disk drive

This comes from Jim Uzzell of Austin, Texas. He writes:

Because the hard disk version of MDOS has not been finished, I took the (KISS) approach to set up my hard drive.

To have AUTOEXEC auto-boot, you must use a sector editor to change (DSK1) AUTOEXEC in sectors 00EE and 00EF of MDOS V1.14 to hdS1. Make this change to a copy of MDOS, of course, and not the original.

You will notice in the following AUTOEXEC file that I have assigned F=WDS1. This allows you to have modules to load to their option menus (see coding for file E—TI XBASIC).

AUTOEXEC FILE

```
AUTOEXEC
ECHO ON
ASSIGN E=hdS1:
ASSIGN F=WDS1:
E:
TYPE E:MENU
```

MENU FILE

```
MAIN MENU
&A....MYWORD
&B....TELCO
&C....EA
&D....ADVBASIC
&E....TI XBASIC
&F....MULTIPLAN
&G....UTILITIES
&H....GPL
ENTER (&) and LETTER
```

Here are paths used on some files:

File A. MY-Word

```
TIMODE
E:GPL.GPL F:DSK1.MW
```

DSK1 is a subdirectory.

File E. TI-XBASIC

```
TIMODE
ASSIGN C=DSK5:
RAMDISK 120
E:GPL.GPL F:XB.EXB
```

File F. MULTIPLAN

```
TIMODE
E:GPL.GPL F:DSK.TIMP.MP
```

File H. GPL

This file is set up so that I can load modules that I don't have on the hard drive.

All the above files (A-H and Menu) and AUTOEXEC are located in the root directory.

The reason I have chosen this approach is tht the current version of MDM5 formats the hard drive as WDS1 and as the development of the Geneve has gone, I probably will have to reformat my hard drive when the final version of the hard disk version of MDOS and disk manager are released.

NX-1000 modification for Disk Label II

This comes from Rodney Wright of Musquodoboit Harbor, Nova Scotia. He writes:

Once again Ed Machonis has written a very useful utility that he has shared with us (Disk Label II, February 1989, Page 34).

However, owners of the Star NX-1000 multifont printer may have a slight problem with the program if they try to print more than one label at a time. The name header does not print out in expanded form the second time around. The "fix" is quite simple.

List the program and fine line No. 6. Now go into edit mode and immediately after the first colon (the one after PRINT #1), type in E$&"@";.

Here is what the line should look like:

```
6 PRINT #1:E$&"@";E$&"E";E$&"G";E$&"-1";E$&"W1";TAB((18-LEN(D$))/2);D$;TAB(18);"";TAB((18-LEN(C$))/2);C$;TAB(18);"";E$&"F";CHR$(15);TAB(2);T$;
```

What you have done is to reset the printer codes for each repetition of your label.

Routine performs modulo division

This comes from Bob Keahey of Albuquerque, New Mexico. He writes:

This subprogram performs modulo division, which is integer division which returns the remainder. I wrote it because I was converting an Microsoft BASIC program to my TI and needed to use modulo division.

```
1 ! LINES 10-50 NOT PART OF SUB !088
2 ! MSBASIC COMMAND Z=X MOD Y !076
10 CALL CLEAR !209
20 INPUT X,Y !246
25 DVD=X :: DVSR=Y !204
30 CALL MOD(DVD,DVSR,RMD)!027
40 Z=RMD !251
45 PRINT Z !246
50 GOTO 20 !099
10000 SUB MOD(DVD,DVSR,RMD)!031
10001 !MODULO DIVISION !216
10010 !BY BOB KEAHEY !232
10020 !DVD IS DIVIDEND !132
10030 !DVSR IS DIVISOR !190
10040 !RMD IS REMAINDER !217
10060 TDD=INT(DVD):: TDR=INT(DVSR)!089
10070 TD1=DVD-TDD :: TD2=DVSR-TDR !248
10079 !THE NEXT TWO LINES DETERMINE THE ACCURACY OF THE ROUNDING PROCESS !028
10080 IF TD1>.49999999 THEN TDD=TDD+1 !176
10090 IF TD2>.49999999 THEN TDR=TDR+1 !205
10100 DVD=TDD :: DVSR=TDR !225
10110 ZT=DVD/DVSR !077
10120 IF ZT<1 THEN RMD=DVD :: GOTO 10170 !204
10130 RMD=DVD !127
10140 FOR L=1 TO INT(ZT)!043
10150 RMD=RMD-DVSR !133
10160 NEXT L !226
10170 SUBEND !168
```

Fix for Archiver 3 to work with Geneve

Barry Boone, author of Archiver, says the following change to the Archiver 3 will make it compatible with the Geneve. Versions of the program dated April 12 or earlier require the change.

Using a sector editor, locate the following string: 04E08C00.

Replace it with: D8018C00.

(See Page 45)

User Notes

Program lets synthesizer read TI-Writer files

This comes from Walter Chmara of Bensalem, Pennsylvania. He writes:

The following short program enables the speech synthesizer to verbalize TI-Writer files. The only glitch in it that I have found is that the Terminal Emulator II module has a problem reading quotation marks. Use the Replace String function to change any quotation marks in the file to a @ symbol before saving it to disk. The program will then have no problem reading it from the disk.

Also, you can add pitch and slope numbers and inflection symbols via TI-Writer into the same file for dramatic effects. You may even misspell some words for a particular pronunciation or to imitate foreign accents. Of course, once you've gone that far, you'll probably want to give it a different SAVE name from the file you for outputting to a printer. And if you intend to create a diskfull of bedtime stories to put yourself to sleep at night, this program can be easily modified to read additional files in sequence.

```
100 REM **TI-READER**   *by
W. Chmara* !127
110 REM Requires Speech Synt
hesizer, TE11, TI-Writer fil
e on disk to read. !201
120 REM Make sure all quotat
ion marks (") in the file ha
ve been changed to @(the "at
" symbol). !215
130 CALL CLEAR !209
140 CALL CHAR(64,"00282828")
!072
150 OPEN #1:"SPEECH",OUTPUT
!122
160 OPEN #2:"DSK1.EXAMPLE",D
ISPLAY ,INPUT ,VARIABLE 80 !
222
170 IF EOF(2)THEN 320 !112
180 INPUT #2:X$,!110
190 PRINT X$ !024
200 B$="" !235
210 FOR A=1 TO LEN(X$)!244
220 A$=SEG$(X$,A,1)!133
230 Y=ASC(A$)!197
240 IF Y=64 THEN 250 ELSE 26
0 !145
250 Y=Y-32 !097
260 IF Y>96 THEN 270 ELSE 28
0 !193
270 Y=Y-32 !097
280 B$=B$&CHR$(Y)!222
290 NEXT A !215
300 PRINT #1:B$ !174
310 GOTO 170 !249
320 CLOSE #2 !152
330 PRINT #1:"//50 160" !145
340 PRINT #1:"THIS IS THE ^
END OF THIS FILE" !228
350 CLOSE #1 !151
360 STOP !152
370 REM "DSK1.EXAMPLE" shoul
d be changed to whatever fil
ename you want to read. !249
```

Routine creates instructions for programs

This comes from Jim Peterson of Tigercub Software. He writes:

How often have you downloaded a program, and had to figure out for yourself how to run it? So, before you pass it on to someone else, why not add some instructions to it? This little routine makes it easy.

To use the program, type the instructions and format them, centered or hyphenated or right-adjusted, just as you want them to appear on screen, and enter each line. It will be written to a D/V163 file named @DATA. When finished, enter END.

Then enter NEW, then MERGE DSK1.@DATA, and RUN to see if everything is okay. If so, load the program needing instructions, make sure its lowest line number is more than 8 and the highest is less than 30721, and enter MERGE DSK1.@DATA.

This is a program that writes a program, so it must be assembled in parts. First key in this part and save it as SHELL.

```
100 CALL CLEAR :: OPEN #1:"D
SK1.@DATA",VARIABLE 163,OUTP
UT :: DEF L$(X)=CHR$(120)&CH
R$(X)!177
110 L=L+1 :: X=X+1 :: ACCEPT
 AT(L,0):M$ :: IF L=24 THEN
CALL CLEAR :: L=0 !163
120 IF M$<>"END" THEN PRINT
#1:L$(X)&CHR$(147)&CHR$(199)
&CHR$(LEN(M$))&M$&CHR$(0)::
GOTO 110 !023
130 REM!154
140 PRINT #1:CHR$(0)&CHR$(4)
&"T@"&CHR$(190)&CHR$(200)&CH
R$(LEN(STR$(X-1)))&STR$(X-1)
&CHR$(0)!234
150 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1 !109
```

Next, key in this part and save it as DSK1.D/MERGE,MERGE.

```
1 DISPLAY AT(12,1)ERASE ALL:
"SKIP INSTRUCTIONS? Y" :: AC
CEPT AT(12,20)SIZE(-1)VALIDA
TE("YNyn"):@Q$ :: IF @Q$="Y"
 OR @Q$="y" THEN 8 !098
2 DISPLAY AT(24,5)ERASE ALL:
"PRESS ANY KEY" !226
3 RESTORE 30721 !214
4 REM!154
5 FOR J@=1 TO T@ :: READ @$
:: DISPLAY AT(J@,1):@$:" " !
254
6 CALL KEY(0,K@,S@):: IF S@=
0 THEN 6 !010
7 NEXT J@ !032
8 REM!154
```

Finally, key in this conversion routine.

```
100 OPEN #1:"DSK1.D/MERGE",V
ARIABLE 163,INPUT :: OPEN #2
:"DSK1.D/MERGE2",VARIABLE 16
3,OUTPUT :: L=129 :: FOR J=1
 TO 8 !244
110 LINPUT #1:M$ :: PRINT #2
:CHR$(0)&CHR$(L+J)&CHR$(156)
&CHR$(253)&CHR$(200)&CHR$(1)
&"1"&CHR$(181)&CHR$(199)&CHR
$(LEN(M$))&M$&CHR$(0):: NEXT
 J !214
120 CLOSE #1 :: PRINT #2:CHR
$(255)&CHR$(255):: CLOSE #2
!136
```

Run that last one to convert the D/MERGE file to another file called D/MERGE2. Then load the SHELL, enter MERGE DSK1.D/MERGE2, run the program and key in the second and third paragraphs above to make it write itself a screen of instructions.

Readers who don't have time to key this in may send $3 for a diskfull of this and other utility programs from Peterson at 156

(See Page 46)

User Notes

(Continued from Page 45)

Collingwood Ave., Whitehall, OH 43213.

Pascal quirk corrected

This comes from Denver Earl Sullivan of Osgood, Indiana. He writes:

Many TI users have abandoned the UCSD p-System, but it is an excellent programming environment once one overcomes the defect in the SYSTEM.LIBRARY file.

If you put the Compiler in #5, the Editor/Filer in #4, and the storage disk in #9, when using the USING SPEECH library routines, as well as any other library routines, you will receive an "Cannot Access Library Files" error when the program is in the compilation stage of development.

To correct this, copy the following files to *each* storage diskette that you are using and place the disk in #4, then the compiler, editor-filer, etc. can reside in any other drive.

Note: This is not necessary for the hard disk version.

ED-FILR:SYSTEM.SYNTAX
COMPILR:SYSTEM.LIBRARY

Classified

SOFTWARE

"BOOT PROGRAM"

Copyrighted by the Miami Users Group Feb. 1989. Not available from any other source or Mail Order Co.

Latest up-to-date version by the original author, John Johnson.

"BOOT" is in assembly language and uses the Horizon RAM Disk "MENU" program, without a RAM Disk.

You will be able to:

1. View a file
2. Catalog a disk
3. Run a program, XB, EA, or cartridges.
4. Access any program on any drive
5. Even run XB programs over 42 sectors.
6. Print any file or disk catalog.

Send $7.95 to:

MIAMI USERS GROUP
YVETTE MCKENZIE
6755 TAMIAMI CANAL ROAD
MIAMI, FL 33126

v6n6

1989 TI FAIRS

MARCH

West Coast Computer Fair, March 17-19, Brooks Hall, San Francisco. San Francisco 99ers to be at Booth 733. For further information, write San Francisco 99ers, 24816 Mango St., Hayward CA 94545.

TICOFF (TI Computer Owners Fun Faire), March 18, Roselle Park High School, Roselle Park, New Jersey. For information, write TICOFF'89 c/o Roselle Park High School, 185 West Webster Ave., Roselle Park, NJ 07204, or call Robert Guellnitz at (201) 241-4550 or (201) 382-5963 or the TICOFF BBS, (201) 241-8902.

APRIL

Fourth Annual New England TI Fayuh, 10 a.m.-5 p.m. April 1, Ramada Inn of IH95 in Woburn, Massachusetts. For information, contact the Boston Computer Society TI99/4A User Group, One Center Plaza, Boston MA 02108.

Alberta TI-Orphan Reunion, April 29 at Innisfail Country Lodge, Innisfail, Alberta, Canada. For information, contact Fred Kessler, Box 20, Sundre, Alberta, Canada T0M 1X0 or (403) 638-3916.

4th Annual Ottawa TI-FEST, April 29 at Merivale High School in Nepean, Ontario, Canada. For information, contact Jane Laflamme, 5480 Canotek Rd. Unit #10, Gloucester, Ontario, Canada K1J 9H6 or (613) 745-2225.

MAY

Multi User Group Conference May 20, Reed Hall/Student Activities Building, Ohio State University, Lima, Ohio. For further information write Lima Users Group, P.O. Box 647, Venedocia, OH 45894, or call Dave Szipple evenings at (419) 228-7109.

JUNE

TI99/4A Users Group (.U.K.) Annual Meeting June 17 in Romley, England. For information, contact Stephen Shaw, 10 Alstone Rd., Stockport, Cheshire, England SK4 5AH.

SEPTEMBER

TI International Expo 89 Sept. 16 at Howard Johnson Inn, 5821 Richmond Highway, Alexandria, Virginia. For further information write Mid-Atlantic Ninety-Nners, TI International Expo 89, P.O. Box 4005, Rockville, MD 20850, (301) 340-7179; or Delphi TI-NET, Teledata; or CompuServe, 74405,1207.

Fourth Annual TI99/4A Seattle Convention, Sept. 23-24 at Kenmore Flea Market in Kenmore, Washington. For further information contact Barb Wiederhold, (206) 361-0799 (voice) or (206) 361-0895.

OCTOBER

Fourth European Tref, begins at 10 a.m. Oct. 7 at Kolpinghuis, Nijmegen, The Netherlands. For information, contact Berry Harmsen, 1e, Oosterparstr 141e, 1091 GZ, Amsterdam, Holland.

Australia TI Fair, 2-6 p.m. Oct. 14, Pavilion, Deepdene Park, Whitehorse Rd., Deepdene, Australia. For information contact TI99/4A Users Group — Melbourne Inc., 88 Main St., Blackburn, Victoria 3130, Australia.

3rd International TI-Users Meeting, 10 a.m.-6 p.m. Oct. 15 at Jugenderherberge Duisberg Wedau, Kalkweg 148, 4100 Duisberg 48, West Germany. For information contact TI-99er Workshop Rheinland, Dept. Allgemein & Software, c/o Mike Heuser, Karl-Marx-Allee 18, 5000 Cologne 71, West Germany, or the organizing committee at PCC, TI-Service, c/o Hans Greiffenberg, Großglocknerstr. 45, D-4100 Duisberg 28., West Germany.

Third Annual CPUG Computer Expo, 7 a.m.-2 p.m. Oct. 15 at Carlisle Fairgrounds on Clay Street in Carlisle, Pennsylvania. Sponsored by Central Pennsylvania 99/4A Users Group, co-sponsored by Cumberland County Amateur Radio Service and 6th Annual Cumberland County Hamfest. For information, contact Central Pennsylvania 99/4A Users Group, P.O. Box 14126, Harrisburg, PA 17104-0126 or the WIZ/TIB BBS, (717) 657-4992 or 657-4997.

NOVEMBER

Chicago TI-Faire, 9 a.m.-5 p.m. Nov. 4 at Holiday Inn, 3505 Algonquin Rd., Rolling Meadows, Illinois. Social evening Nov. 3, dinner evening of Nov. 4. Sponsored by Chicago Area TI99/4A Users Group. For information contact Sandy Bartels, Chicago Area TI99/4A Users Group, P.O. Box 578341, Chicago, IL 60657 or (312) 859-3850.

Milwaukee TI-Faire, 9 a.m.-5 p.m. Nov. 5 at Quality Inn, 5311 S. Howell Ave., Milwaukee, Wisconsin (across from Mitchell Field Airport). For information call Gene Hitz, 4122 N. Glenway, Milwaukee, WI 53222 or (414) 535-0133.

1990 TI FAIRS

FEBRUARY

TI-Fest West '90, Feb. 17-18, Day's Inn, 88 E. Broadway, Tucson, Arizona. Sponsored by Southwest 99ers. For information, call (602) 747-5046 or the Cactus Patch BBS, (602) 795-1953 or check GEnie. For room reservations, call (602) 791-7581 by Jan. 16 and mention Fest-West.

This TI event listing is a permanent feature of MICROpendium. User groups and others planning events for TI/Geneve users may send information for inclusion in this standing column. Events will remain listed throughout the year for reference for the coming year.

Classified

SOFTWARE

FOR SALE

120 original programs $1 each, 18 full collection disks $5 each, 5 Tips disks reduced to $5 each, 3 Nuts & Bolts disks reduced to $10 each. Send $1 refundable for Tigercub catalog. 250 disks of public domain and fairware, $1.50 each. Send SASE for list or $1 refundable) for catalog, to TI-PD, 156 Collingwood Ave., Whitehall OH 43213. v6n12

CBM-III

Now in V5.6 — a better mousetrap for your finances. More than 50 menu-driven functions (recording, editing, balancing, reconciling, sorting, searching, analysis, etc.) with 32 help screens from a 4-screen menu. On-disk documentation. Epson printer compatible. Geneve compatible. Specify greatest disk drive configuration. $15.00 to: W. Irving Crowley, Lost Canyon Rd., Pine Level, AL 36065. Ph (205) 584-7644. v6n6

APPOINTMENT SCHEDULER
FOR
GENEVE 9640
$15.00

ARTIST UTILITY
FOR
XBASIC
$12.00

DDI SOFTWARE
2004B LEEANN
AUSTIN, TX. 78758 v6n6

SYSTEMS

TI99/4A

Speech Synth, PEB, monitor, Panasonic KX-P1091, XB, TI-Writer, Disk Fixer, Disk Manager, SS, drive, 128K, RS232, 17 cartridges, with desk and printer stand, make offer. K. Marinelli, Beverly, MA (508) 927-6928. v6n6

TI99/4A SYSTEM FOR SALE

TI99/4A, PE-Box, 2 disk drives & controller, RS-232 & cable, 32K card, Extended BASIC, Editor/Assembler, TI-Writer, BRAIN, TI-1-2-3, GRAPHX, C, Pilot, Forth Compilers. Mint condition, $275 or best offer. Check, Vista, MasterCharge. Julian Achim, 8425 68th Way North, Pinellas Park, FL 34665. (813) 545-4354. v6n6

Policy

The cost of classified advertising is 25 cents per word. Classified display (i.e., special formatting or graphics) is $9 per column inch. Classified advertisements must be paid in advance. Classified advertisers may request a category under which they would like their advertisements to appear, but the final placement decision is the responsibility of the publisher.

Classified deadlines will be kept open for as long as practical. For the purpose of classified advertising deadlines, any classified ad received later than the first day of any month cannot be assured of placement in the next edition. We will do our best to include every advertisement that is submitted in the earliest possible edition.

The publisher offers no guarantee that any advertisement will be published in any particular issue. Any damages that result either from errors in copy or for failure to be included in any particular edition will be limited to the amount of the cost of the advertisement itself. The publisher reserves the right to reject any advertisement.

The advertiser may elect to publish the advertisement in subsequent editions at the same charge, payable prior to publication. The deadline for carryover classifoeds is the same as for new advertising.

In submitting an ad, please indicate whether you would like a refund if it is not published in the requested edition or whether you would like us to hold it for the next edition. Cancellations and refunds cannot be made after the second day of the month.

Send classified advertising to: MICROpendium, P.O. Box 1343, Round Rock, TX 78680.

MISCELLANEOUS

Texas Instruments
TI-99/4a
Hardware & Software

—TURBO-PASC'99 (Pascal)! $59.95 E/A required.— Proto-Board for PEB by Willforth only $35. Joy-Sticks $10.00 XB Detective $10.00. Rave '99 KeyBoard!! $199. Speech Editor $20. Printer Cable (PIO) $24. Jungle Hunt, Protector II, Moon Patrol, PacMan $15 ea. TI Dust Cover $4, Adventure $10.00. Adv Hint Book $8. TI Extended BASIC $40, P-Code Card w/software $159. TI-Artist $20. Princess & Frog, AntEater, HenPecked $10 ea.

*TI-BASE $25 (great) Data Base, PEB Hardware Manual 32K, Disk Ctrl, RS232, RAMdisk Cards, Cables & Connectors

L.L. CONNER ENTERPRISE
COMPUTER&ELECTRONICS
1521 Ferry Street
Lafayette, IN 47904

CALL 317-742-8146 or CALL 317-423-4879 to order UPS today!
Visa or M/C Accepted, add $3 shipping v6n6

GAMES! EDUCATIONAL! HARDWARE! —TI99/4A
CALL OR WRITE FOR FREE CATALOG:
JOY ELECTRONICS, INC; P.O. BOX 524526;
DALLAS, TEXAS 75354-2526;
(214) 243-5371, LOCAL; (800) 442-3892, TEXAS;
(800) 527-7438, OUTSIDE TEXAS v6n6

MISCELLANEOUS

PASCAL COMPLETE SYSTEM	$150
BARCODE READER	$ 30
FULL TI PE/BOX	$350
EMPTY TI PE/BOX	$135
18" P-BOX EXTENSION CABLE	$ 25
SCM DOT MATRIX PRINTER	$110
SPEECH SYNTHESIZER used	$ 45
PARALLEL PRINTER CABLE 6'	$ 20
RS232 "Y" CABLE	$ 15
STAND ALONE DISK DRIVE "TI"	$ 79
PE-BOX TECH TRAINING MANUAL	$ 30
TI ORIGINAL COLOR MONITOR	$175
SERVICE MANUAL (CONSOLE/P BOX)	$ 25
4A FACTORY REPAIR MANUAL	$ 30
EXTENDED BASIC used w/new book	$ 35

PLUS 5% SHIPPING CALL OR WRITE for complete free list
JIM LESHER, 722 HUNTLEY
DALLAS, TEXAS 75214, 214 821 9274 v6,n6

ENORMOUS INVENTORY

TI 99/4A. Laser,
Apple & IBM Compatibles.

"Some Commodore and Atari Items. (No Catalog on Above Two Systems.)" Full Hardware & Software on All Other Systems. Complete Line of Computer Forms.

BRAATZS
Computer Services

719 E. Byrd St., Appleton, WI 54911. **Catalogs** $2. MC/Visa accepted. 414-731-3478 (ORDER LINE ONLY) (Call 414-731-4320 after 6:00 P.M. Wisconsin time.) Custom made covers (send sizes). v6n6

The LEADING monthly devoted to the TI99/4A

Subscription Fees

$20 for 12 issues via domestic second class mail
$25.25 (U.S. funds) Mexican delivery
$27.50 (U.S. funds) Canadian delivery
$25.00 (U.S. funds) for 12 issues other foreign delivery via surface mail
$37.00 (U.S. funds) for 12 issues other foreign delivery via air mail

Outside U.S., pay via postal or international money order or credit card; personal checks from non-U.S. banks will be returned

Address Changes

Subscribers who move may have the delivery of their most recent issue(s) delayed unless MICROpendium is notified six weeks in advance of address changes. Please include your old address as it appears on your mailing label when making an address change.

Back Issue Policy

Back issues of MICROpendium are available to subcribers only. Those wishing back issues may notify us of the issue(s) desired and include $2 per issue desired in a check or money order or by credit card. (Minimum credit card order is $9.) No shipping charge in U.S. and Mexico; Texas residents add 7.5% sales tax. Shipping charge of 30 cents per issue to Canada. For other foreign delivery, add 50 cents per issue surface mail, $2 per issue air mail. No discounts on orders of sets. All prices U.S. funds.

OUT OF STOCK: Vol 1, nos. 1-2, Vol 2, no. 1

Miscellany

TI-Forth Disks (2 disks, program and demo disks, no manual).....$6.00
MICROpendium Index (2-SSSD disks, XBASIC required).....$5.00
Disk of programs from one issue of MICROpendium (must be a subscriber to order).....$4.00
12 monthly disks of programs appearing in each edition of MICROpendium (must be a subscriber to order/starts with April 1989).....$40.00
Magazine holders (12/set-add $1 shipping/order).....$3.00

Send name, address, product(s) ordered, check, money order or Visa/MasterCard number and expiration date ($9 minimum on credit card orders $9) to: MICROpendium, P.O. Box 1343, Round Rock, TX 78680. (Foreign orders write for postage fees.)

Tell us about it

Please let us what columns or features you like the most about MICROpendium. Rank your selections in order of preference using this form. Return it to us when you renew your subscription.

1. _____
2. _____
3. _____
4. _____

Other suggestions:_____

_____

_____

Send me the next 12 issues of MICROpendium. I am enclosing $ in a check or money order in U.S. funds. Or bill my ☐ ☐ (Check one)

Exp. Date _____

Card No. _____

Minimum credit card order is $9

Signature_____

(required on credit card orders)

Mail to: MICROpendium, P.O. Box 1343, Round Rock, TX 78680

Name_____

Address_____

City_____

State____**ZIP**_____

The numbers on the left of your mailing label indicates the cover date of the last issue on your subscription. v6,n5

SECOND CLASS